04.22.05

AF 1631

Express Mailing Label No.: EV 524793705 US
IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of Atty. Docket No.: 2551-026
Douglas M. BLAIR
Appln. No.: 09/881,234 Group Art Unit: 1631
Filing Date: Jun. 14, 2001 Examiner: Smith, C.

For: **APPARATUS AND METHOD FOR PROVIDING SEQUENCE DATABASE COMPARISON**

* * * * * * * * * * * * * * * *

NOTICE OF APPEAL AND APPEAL BRIEF TRANSMITTAL

* * * * * * * * * * * * * * * *

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Enclosed, please find:

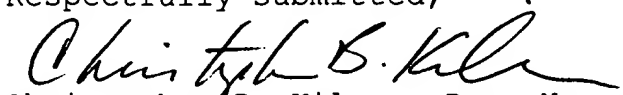
1. Notice of Appeal (to reinstate prior appeal); and

2. Appellants' Brief on Appeal in response to the Final Rejection dated January 21, 2005, with Claim and Evidence Appendices.

No fees are believed to be due in connection with this paper since the fee for the Notice of Appeal was previously filed on December 9, 2003 and the fee for the Brief on Appeal was previously filed on March 9, 2004.

However, in the event that it is determined that the payment of the fee is required for consideration of this paper, the Commissioner for Patents is hereby authorized to charge all necessary fees to the Deposit Account No. 18-1579. A duplicate copy of this letter is enclosed.

Respectfully submitted,



Christopher B. Kilner, Reg. No. 45,381
Roberts Abokhair & Mardula, LLC
11800 Sunrise Valley Dr.,Suite 1000
Reston, VA 20191
(703) 391-2900

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of Atty. Docket No.: 2551-026

Douglas M. BLAIR

Appln. No.: 09/881,234 Group Art Unit: 1631

Filing Date: June 14, 2001 Examiner: Smith, C.

For: **APPARATUS AND METHOD FOR PROVIDING SEQUENCE DATABASE COMPARISON**

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

* * * * *

NOTICE OF APPEAL FROM THE PRIMARY EXAMINER TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

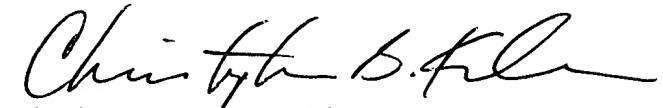
* * * * *

Dear Sir:

Applicant hereby reinstates the previously filed appeal to the Board of Patent Appeals and Interferences from the decision of the Primary Examiner, mailed January 21, 2005, finally rejecting claims 1-23.

Applicant respectfully submits that no fee is necessary since Applicant previously paid for the Notice of Appeal filed December 9, 2003.

Respectfully submitted,



Christopher B. Kilner, Esq.
Registration No. 45,381
Roberts Abokhair & Mardula, LLC
11800 Sunrise Valley Drive,
Suite 1000
Reston, VA 20191-5302
(703) 391-2900

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of Atty. Docket No.: 2551-026

Douglas M. BLAIR

Appln. No.: 09/881,234 Group Art Unit: 1631

Filing Date: Jun. 14, 2001 Examiner: Smith, C.

For: **APPARATUS AND METHOD FOR PROVIDING SEQUENCE DATABASE COMPARISON**

* *

APPELLANT'S BRIEF ON APPEAL UNDER 37 C.F.R. § 41.47

* *

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In accordance with the provisions of 37 C.F.R. § 41.37, Appellant submits the following:

I. REAL PARTY IN INTEREST

Based on information supplied by Appellants, and to the best of Appellants' legal representatives' knowledge, the real party in interest is Parabon Computation, Inc.

II. RELATED APPEALS AND INTERFERENCES

Appellants, as well as Appellants' assigns and legal representatives are unaware of any appeals or interferences which will be directly affected by, or which will directly affect, or have a bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 1-23 are currently pending. No claims have been allowed. No claims have been canceled. Claims 1-23 are appealed. Claims 1-23, as amended with the original Appeal Brief of March 9, 2004, are set forth in the Appendix.

IV. STATUS OF AMENDMENTS

An amendment was filed with the original Appeal Brief of March 9, 2004 to eliminate the alleged indefiniteness of the abbreviations CPU, ID, and BLAST from the claims so as to place the claims in better condition for appeal. The only other amendments in the application, filed November 22, 2002 and July 14, 2003 to amend the specification, were entered.

V. SUMMARY OF THE INVENTION

Appellants' disclosed and claimed invention is directed to a method and system of comparing a query and a subject database using a distributed computing platform. The databases are divided into data elements having a size within a specified range. All data elements and task definitions are sent to a master CPU of a master-slave distributed computing platform, wherein task definitions

comprise at least one comparison parameter, at least one executable comparison element, and a query and a subject data element ID/descriptor. Data elements are sent alternately from query and subject data elements. A task definition is sent for each task from the master CPU to one of a plurality of slave CPUs when all parts of a task definition and data elements referenced by the task definition are available at the master CPU. Data elements are then sent to the slave CPUs for performance of the tasks. Task results for each task are returned to a CPU.

In one of its broadest embodiments, the claimed invention is drawn to (claim 1) a method of comparing a query dataset N with a subject dataset M, comprising: dividing said query dataset N into n_N data elements having a size within a specified range and dividing said subject dataset M into n_M data elements having a size within said specified range (see figs. 1B, 3, and box 620 of figure 6; paras. [68] and [87] of the specification); determining a number of tasks for an entire comparison of datasets N and M as $n_N \times n_M$ (see box 628 of fig. 6; paras. [69] and [90] of the specification); sending all data elements and task definitions to a master CPU of a master-slave distributed computing platform, wherein task definitions comprise at least one comparison parameter, at least one executable element capable of performing comparisons, a query data element ID/descriptor, and a subject data element ID/descriptor, and wherein data elements are sent alternately from query and subject data elements (see computer topology of fig. 5 and boxes 630-640 of fig. 6; paras. [70] and [90]-[92] of the specification); sending a task definition for each task from the master CPU to one of

a plurality of slave CPUs when all parts of a task definition and data elements referenced by said task definition are available at said master CPU and sending data elements referenced by said task definition to said slave CPU (see box 650 of fig. 6; paras. [71] and [92] of the specification); performing each task on a slave CPU (see box 650 of fig. 6; paras. [71] and [92] of the specification); and returning task results for each task to said master CPU (see box 730 of fig. 7; paras. [72] and [96] of the specification).

In another of its broadest embodiments, the invention is drawn to a system (Claim 13) for comparing a query dataset N with a subject dataset M, comprising: a master CPU of a master-slave distributed computing platform; a plurality of slave CPUs capable of communication with said master CPU; and a client CPU (see fig. 5 and para. [63] of the specification) with instructions for: dividing said query dataset N into n_N data elements having a size within a specified range and dividing said subject dataset M into n_M data elements having a size within said specified range (see figs. 1B, 3, and box 620 of figure 6; paras. [68] and [87] of the specification); determining a number of tasks for an entire comparison of datasets N and M as $n_N \times n_M$ (see box 628 of fig. 6; paras. [69] and [90] of the specification); sending all data elements and task definitions to said master CPU of a master-slave distributed computing platform, wherein task definitions comprise at least one comparison parameter, at least one executable element capable of performing comparisons, a query data element ID/descriptor, and a subject data element ID/descriptor, and wherein data elements are sent

alternately from query and subject data elements (see computer topology of fig. 5 and boxes 630-640 of fig. 6; paras. [70] and [90]-[92] of the specification); said master CPU comprising instructions for: sending a task definition for each task to one of said plurality of slave CPUs when all parts of a task definition and data elements referenced by said task definition are available at said master CPU; and sending data elements referenced by said task definition to said slave CPU (see box 650 of fig. 6; paras. [71] and [92] of the specification); and said slave CPUs including instructions for: performing each task (see box 650 of fig. 6; paras. [71] and [92] of the specification); and returning task results for each task to said master CPU (see box 730 of fig. 7; paras. [72] and [93]-[95] of the specification).

The method of claim 1 and the system of claim 13 can be further limited by steps and means: i) for randomizing sequence order of each dataset if either dataset contains related sequences in a contiguous arrangement (claims 2 and 14, box 600 of fig. 6, and paras. [66] and [87] of the specification); ii) for formatting said datasets so as to use exactly the same ambiguity substitutions (claims 3 and 15, box 610 of fig. 6, and paras. [67] and [87] of the specification); iii) wherein dividing said datasets into data elements further comprises: stripping all metadata from data; packing said data into an efficient structure; creating an index for said data and packing said index and said data in an uncompressed data structure; and compressing said uncompressed data structure into a data element using a redundancy reduction data compression method (claims 4 and 16, boxes 621-626 of fig. 6, and

paragraph [68] of the specification); iv) for sending remaining data elements from a more numerous of said datasets to said master CPU followed by all task definitions for otherwise complete tasks if there are fewer data elements from one dataset (claims 5 and 17, para. [70]); and v) wherein said datasets are selected from the group consisting of genomic and proteomic databases (claims 12 and 23, fig. 2C, paras. [9] and [62] in the specification).

The broad method and system claims can also be further limited, wherein performing a task on said slave CPU further comprises: uncompressing and unpacking data from said query and subject data elements; looping through query sequences from said query data element to perform setup, preprocessing and table generation for each row of comparisons; looping through subject sequences from said subject data element and, for each pair of query and subject sequences, performing a comparison using said executable element and finding results based on said at least one comparison parameter; and storing minimal information that will allow reconstruction of said result (claims 6 and 18, fig. 7, para. [72] of the specification), and in which storing said minimal information can optionally comprise: storing index information for said query and said subject sequence; storing bounds information for start and stop of said query and subject sub sequences; storing data that quantify fulfillment of significance criteria for a significant match; and storing an efficiently encoded representation of alignment between said bounds corresponding to a high-scoring segment pair (claims 7 and 19, figs. 4A and 4B, paras. [72], [85, and

[86] of the specification), or it can further comprise storing a seed point and sum-set membership for each alignment for BLAST (claim 8, para. [72]), or it can further comprise storing task results in a task result file, said file including query and subject sequence data and metadata corresponding to the task that the results came from, metadata for the subject sequence, the partial subject sequence data corresponding to the subject bounds of the significant alignment result, and any other results data for each result in the task results (claims 9 and 20, para. [74]).

The method and system of claims 9 and 20 can further comprise generating a BLAST report for each query data element (claims 10 and 21, box 690 of fig. 6, paras. [74] and [99] of the specification), which can further comprise concatenating results from all BLAST reports to produce a text file identical to a blastall run of said query and subject datasets (claims 11 and 21, box 690 of fig. 6, paras. [74] and [99] of the specification.

VI. ISSUES

The issues on Appeal are:

Grounds 1 - Are claims 4 and 16 indefinite under the second paragraph of 35 U.S.C. § 112 due to the use of the term "efficient structure" in the claims?

Grounds 2 - Are claims 7 and 19 indefinite under the second paragraph of 35 U.S.C. § 112 due to the use of the term "efficiently encoded representation of alignment" in the claims?

Grounds 3 - Is claim 8 indefinite under the second paragraph of 35 U.S.C. § 112 due to the use of the term

"seed point and sum-set membership" in the claims?

Grounds 4 - Are claims 1 and 13, and all the remaining claims dependent thereon, indefinite under the second paragraph of 35 U.S.C. § 112 due to the term "said task definition" lacking a proper antecedent basis in the claims?

Grounds 5 - Are claims 1, 4, 6-13, and 18-23 unpatentable over the publication to Smith et al. (1996) in view of the publication to Altschul et al. (1990) and U.S. Patent No. 5,862,325 to Reed et al. as being obvious?

VII. ARGUMENTS

Claim Rejections - 35 USC §112

Grounds 1, 2, 3, and 4 are related to rejections under 35 USC §112. As previously submitted and cited in M.P.E.P. §2173.01, Appellants submit that a fundamental principle contained in the second paragraph of 35 U.S.C. § 112 is that Appellants are their own lexicographers. Appellants can define in the claims what they regard as their invention essentially in whatever terms they choose so long as the terms are not used in ways that are contrary to accepted meanings in the art. Appellants may use functional language, alternative expressions, negative limitations, or any style of expression or format of claim which makes clear the boundaries of the subject matter for which protection is sought. As noted by the court in *In re Swinehart*, 439 F.2d 210, 160 USPQ 226 (CCPA 1971), a claim may not be rejected solely because of the type of language used to define the subject matter for which patent protection is sought.

Appellants again submit that the proper focus during examination of claims for compliance with the requirement for definiteness of 35 U.S.C. §112, second paragraph as defined in M.P.E.P. §2173.02 is whether the claim meets the threshold requirements of clarity and precision, not whether more suitable language or modes of expression are available. When the Examiner is satisfied that patentable subject matter is disclosed, and it is apparent to the examiner that the claims are directed to such patentable subject matter, he or she should allow claims which define the patentable subject matter with a reasonable degree of particularity and distinctness. *Some latitude in the manner of expression and the aptness of terms should be permitted even though the claim language is not as precise as the examiner might desire.* Examiners are encouraged to suggest claim language to appellants to improve the clarity or precision of the language used, *but should not reject claims or insist on their own preferences if other modes of expression selected by appellants satisfy the statutory requirement.*

The essential inquiry pertaining to this requirement is whether the claims set out and circumscribe a particular subject matter with a reasonable degree of clarity and particularity. Definiteness of claim language must be analyzed, not in a vacuum, but in light of:

(A) The content of the particular application disclosure;

(B) The teachings of the prior art; and

(C) The claim interpretation that would be given by one possessing the ordinary level of skill in the pertinent art at the time the invention was made.

Grounds 1

In the Final Rejection, claims 4 and 16 were rejected as indefinite under the second paragraph of 35 U.S.C. § 112 due to the use of the term "efficient structure" in the claims. Claims 4 and 16 recite the claim limitation of "packing said data into an efficient structure." The Office Action has objected to the term "efficient structure" in this limitation as being allegedly vague and indefinite. To support this position, the Office Action cites to the *Merriam-Webster* dictionary definition of "efficient" as "productive of desired effects" to conclude that "one of skill in the art would question what characteristics must be present for the structure to contain these desired effects."

As previously presented by Appellants, the present specification lists examples of an "efficient structure" in the field of bioinformatics (Appellants have previously directed attention to paragraph 68, wherein the term is defined: "efficient structure, e.g. 2 bits per nucleotide with appropriate encoding, 5 bits per amino acid residue with appropriate encoding, etc."). The term does not need an explicit definition since its meaning is readily understood by one of skill in the art. As a term of art in bioinformatics, the Office Action's citation to *Merriam-Webster* is inappropriate. Indeed, in the background portion of the present specification, Appellants discuss the well known use of NCBI BLAST's *formatdb* program to pack sequence data into one of the natural and commonly used efficient structures (2-bits per nucleotide), disclosed at paragraph 19:

> "The NCBI BLAST suite of programs includes a program called *formatdb* that converts a text-

> based sequence database into a specialized format. The *blastall* program requires such a formatted database for the subject database when it performs a search. The "*blastn*" algorithm implemented in *blastall*, for instance, requires a formatted nucleotide database for the subject database. In order to reduce the size of the database, and as an optimization for speedier comparison of nucleotide sequence data, *formatdb* converts each ASCII character representing a single nucleotide (e.g. 'A', 'C', 'G', 'T', 'U', 'a', 'c', etc.) into a two bit value representing one of the four standard nucleotides, A, C, G or T(U). These are then packed four at a time into 8-bit bytes in a packed database file."

Likewise, the cited art of record to Matsumoto et al. discussing "Biological Sequence Compression Algorthms" states at the bottom of page 43 that "[s]ince DNA sequences contain four symbols, 'a,' 't,' 'g,' and 'c,' if these were totally random, *the most efficient way to represent them would be using two bits for each symbol.*" (Emphasis added)

As noted by the Matsumoto et al. paper, compression schemes exist that take advantage of the non-random nature of DNA to provide further compression. However, by providing examples in the specification to the use of the minimum number of bits needed to express the possibilities, Appellants submit they have provided one of skill in the art with the minimum characteristics required of an "efficient structure," thus circumscribing the claimed metes and bounds, albeit in a broad and flexible manner. In accordance with M.P.E.P. § 2173.04, breadth of a claim is not to be equated with indefiniteness. *In re Miller*, 441 F.2d 689, 169 USPQ 597 (CCPA 1971). If the scope of the subject matter embraced by the claims is clear, and if appellants have not otherwise indicated that they intend

the invention to be of a scope different from that defined in the claims, then the claims comply with 35 U.S.C. 112, second paragraph.

And finally, Altschul et al. *do* in fact suggest that those of skill in the art understand efficient packing since Altschul et al. disclose that it "is advantageous to compress the database by packing 4 nucleotides into a single byte" (p. 405, col. 1), which, because a byte consists of 8 bits, is the same as 2-bits per nucleotide.

In view of the subjective documentary evidence, Appellants submit that the term "efficient structure" as used within the claim term "packing said data into an efficient structure" is definite to one of skill in the art and that claims 4 and 16 comply with the second paragraph of 35 USC 112.

Grounds 2

Claims 7 and 19 were rejected as indefinite under the second paragraph of 35 U.S.C. § 112 due to the use of the word "efficiently" in the term "efficiently encoded representation of alignment."

With respect to the term "efficiently encoded representation of alignment," Appellants submit that the term has been taken out of its full context and that one of ordinary skill in the art of bioinformatics would clearly understand the meaning of the entire term, "an efficiently encoded representation of alignment between said bounds corresponding to a high-scoring segment pair." As discussed above with respect to Grounds 1, efficient encoding entails use of the minimum number of bits needed to represent the data and, as previously submitted to the Examiner, BLAST uses a specific data format to represent alignment pairs,

such that the term is reasonably clear to one of ordinary skill in the art.

Indeed, as with claims 4 and 16 discussed in Grounds 1 above, the Office Action alleges that Appellants "have not shown where in the specification [using the minimum number of bits needed to represent the data] is explained or sufficiently proven (via documentation) that this definition of the phrase is well known in the art."

In response, the Appellants again submit that the well known *formatdb* program which was disclosed as follows in the specification:

> "to reduce the size of the database, and as an optimization for speedier comparison of nucleotide sequence data, *formatdb* converts each ASCII character representing a single nucleotide (e.g. 'A', 'C', 'G', 'T', 'U', 'a', 'c', etc.) into a two bit value representing one of the four standard nucleotides, A, C, G or T(U),"

as discussed in paragraph 19 uses efficient encoding. Furthermore, the disclosure of Matsumoto et al. states that:

> "[s]ince DNA sequences contain four symbols, 'a,' 't,' 'g,' and 'c,' if these were totally random, the *most efficient way to represent them* would be using two bits for each symbol"

and thus sufficiently demonstrates that one of skill in the art understands what is meant by "efficiently encoded" sequence data.

Furthermore, additional evidence that the concept of efficient encoding is well understood in the art of bioinformatics was previously submitted to the Patent Office. Varre et al. is a 1999 article published in *Bioinformatics* and, with respect to encoding scripts for sequence comparison, discloses:

> "To be comparable, descriptions have to be written in the same language. We use binary language *because efficient encoding procedures are known*. As DNA is made up from 4 ($=2^2$) possible bases, each of them might be encoded over 2 (the exponent) bits. A *n*-bases long sequence is thus encoded over 2*n* bits." (Emphasis added)

In view of the subjective documentary evidence, Appellants submit that the term "efficiently encoded representation of alignment" is definite to one of skill in the art and that claims 7 and 19 comply with the second paragraph of 35 USC 112.

Grounds 3

Claim 8 was rejected as indefinite under the second paragraph of 35 U.S.C. § 112 due to the use of the term "seed point and sum-set membership" in the claim. The Office Action further alleges that "seed point and sum set membership" is vague and indefinite since "it is unclear how the Appellants intend this phrase to be defined." In rejecting Appellants arguments, the Office Action alleged that Appellants' prior arguments were not supported by documentation and the terms were not used by Altschul et al.'s original BLAST paper.

In response, Appellants agree with the implicit tenets of the Office Action that one of ordinary skill in the art would be aware of the work of Altschul et al. and would be familiar with NCBI's BLAST program. Indeed, as shown in the attached Karlin et al. reference, Altschul and Karlin discussed the concept of "sum" statistics in 1993 to address multiple high-scoring segments pairs (HSPs) in molecular sequences that can occur due to gaps in "consistently ordered segment pairs in sequence alignments." The present invention discloses the use of

BLAST 2 (2.0.14), which has sum statistics output enabled as a default, as shown in the "Search Strategy" portion of the previously submitted BLAST Help Manual, wherein it says:

> "By default the programs use 'Sum' statistics (Karlin and Altschul, 1993). As such, the statistical significance ascribed to a set of HSPs may be higher than that ascribed to any individual member of the set. Only when the ascribed significance satisfies the user-selectable threshold (E parameter) will the match be reported to the user."

Since the sum statistics for an alignment relate to a set of HSPs comprised of various members, the identified HSPs were referred to by the present inventors and others in the field as the "sum set" and membership of an HSP therein as "sum set membership." The terms that appear to be the most often used in the art are "*set of HSPs*," as in the above quote from the BLAST Help Manual, and "*sum group*," as used in note 6 of the Release History section of the BLAST 2.0 Release Notes, also previously sumbitted. However, the present application's use of "sum set" is well understood in the art to be synonymous with "sum group" and "set of HSPs" and therefore is sufficiently definite to one of skill in the art.

Likewise, "seed point" is well understood by those of skill in the art. As stated in Karlin et al. in the second paragraph under the heading "The Construction And Statistical Evaluation Of Gapped Local Alignments," a seed is a single aligned pair - "Starting from a single aligned pair of residues, called the *seed*, the dynamic programming proceeds..." If the sequences being compared are known, the most efficient way to identify the seed is to identify the

point at which it occurs in the sequence, that is the *seed point*.

Indeed, the term "seed point and sum set membership" refers to the minimum information required to reconstruct (on the client side) multiple gapped alignments, and group the ones together which give the smallest resulting summed e-value (in accordance with Karlin et al.). That is, it identifies the location of each alignment (the seed point) and the HSPs that go together as "summed" alignments (the sum set membership). In view of this, Appellants submit that the term is sufficiently definite to one of skill in the art.

Further, in response to the Office Action's statement that the term was not used in the cited BLAST prior art, such as the cited 1990 reference to Altschul et al., Appellants point out that the statistical approach of Sum statistics was not part of the original BLAST and was first introduced by Karlin and Altschul in 1993, as noted above. However, the cited prior art to Anderson et al. further demonstrates the use of the term "seed" as a term of art to refer to the sequence S(0) chosen for comparison (see, e.g., pages 350-352). Additional evidence that "seed" is a term of art used in this manner was previously submitted in the form of the article by Brudno et al., which on page 2 states that the sequence comparison algorithm "works by chaining together pairs of similar regions, one from each of the two input DNA sequences; we call such pairs of regions *seeds*. More precisely, a seed is a pair of words of length k with at least n identical base pairs."

In view of the subjective documentary evidence, Appellants submit that the term "seed point and sum set

membership" is definite to one of skill in the art and that claim 8 complies with the second paragraph of 35 USC 112.

Grounds 4

Claims 1 and 13 were rejected as indefinite under the second paragraph of 35 U.S.C. § 112 due to the term "said task definition" lacking a proper antecedent basis in the claims.

Appellants admit that the claim language is not necessarily as clear as it could be. When practicing the invention, multiple task definitions are sent to multiple CPUs, but each task definition has the same properties and is sent only when the data elements it references are available. Despite variously using "tasks," "task definitions" (2X), "a task definition for each task," and "all parts of a task definition" prior to claiming "said task definition," Appellants submit that the claim language is sufficiently clear to refer to the immediately preceding form of the term "all parts of a task definition" as used in the term "all parts of a task definition and data elements referenced by said task definition." Even though the claim language may not be as precise as the examiner might desire, the examiner never suggested claim language to Appellants to improve the clarity or precision of the language used, but instead improperly rejected the claims despite the mode of expression selected by Appellants satisfying the statutory requirement, in direct contravention of M.P.E.P. § 2173.02.

Indeed, under M.P.E.P. § 2173.05(e), a claim is indefinite when it contains words or phrases whose meaning is unclear. The examples in the M.P.E.P. suggest that a

lack of clarity could arise where a claim refers to "said lever" or "the lever," where the claim contains no earlier recitation or limitation of a lever and where it would be unclear as to what element the limitation was making reference. Similarly, if two different levers are recited earlier in the claim, the recitation of "said lever" in the same or subsequent claim would be unclear where it is uncertain which of the two levers was intended. No such problems exist in the present claims; "a task definition" is claimed prior to "said task definition" and repeated use of "a task definition" in the phrase "sending a task definition for each task from the master CPU to one of a plurality of slave CPUs when all parts of a task definition and data elements referenced by said task definition are available at said master CPU" does not suggest that the repeated use of "a task definition" refers to different task definitions.

In view of this, Appellants submit that the scope of the claims would be reasonably ascertainable by those skilled in the art, and therefore the claims are not indefinite. *Ex parte Porter*, 25 USPQ2d 1144, 1145 (Bd. Pat. App. & Inter. 1992) ("controlled stream of fluid" provided reasonable antecedent basis for "the controlled fluid").

In view of the above-cited reasons, Appellants submit that claims 1-23 are definite and respectfully request reconsideration and withdrawal of the rejections. Appellants further note that claims 2-3, 5, and 14-17 have not been rejected in view of the prior art and are thus admittedly allowable upon being found definite.

Claim Rejections - 35 USC §103

Grounds 5

Claims 1, 4, 6-13, and 18-23 were rejected under 35 U.S.C. 103 are being obvious over the publication to Smith et al. (1996) in view of the publication to Altschul et al. (1990) and U.S. Patent No. 5,862,325 to Reed et al.

To establish a *prima facie* case of obviousness, three basic criteria must be met (See M.P.E.P. Section 2143). First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988); *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992).

Second, there must be a reasonable expectation of success. This requirement is primarily concerned with less predictable arts, such as the chemical arts.

Finally, the prior art must teach or suggest each and every limitation of the claimed invention, as the invention must be considered as a whole. *In re Hirao*, 535 F.2d 67, 190 U.S.P.Q. 15 (C.C.P.A. 1976).

The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, not in Apellant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991).

No Motivation to Combine

In the present case, none of these criteria have been met in the Office Action. First, there is no suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in

the art, to modify the search launcher interface of Smith et al. or combine it with Altschul et al. and Reed et al.

M.P.E.P. 2141.02 requires that an invention be considered as a whole. The present invention, *as a whole*, is drawn to a method or system for comparing a query dataset N to a subject dataset M using not only a network, but a *distributed computing platform*. A client computer in the claimed system and method divides the query dataset N into n_N data elements having a size within a specified range, divides the subject dataset M into n_M data elements having a size within said specified range, and determines a number of tasks for an entire comparison of datasets N and M as $n_N \times n_M$. The client computer then sends all data elements and task definitions to a master CPU of a master-slave distributed computing platform, and the master CPU sends a task definition and its associated data elements for each task to one of a plurality of slave CPUs of the distributed computing platform. The slave CPUs of the distributed computing platform perform the tasks (inherently in parallel) and return the results to the master CPU.

In contrast, *none* of Smith et al., Altschul et al. or Reed et al. even mentions distributed computing. In making the rejection, the Office Action erroneously looks to *Merriam-Webster* for the definition of "system" instead of looking to the broadest reasonable interpretation *consistent with the specification* as required by M.P.E.P. 2111. *Merriam-Webster* for the definition of "system" is not consistent with the distributed computing platform disclosed in the specification.

M.P.E.P. 2141.02 further requires that the prior art be considered as a whole, including portions that teach away from the invention. Smith et al., as a whole, *teaches against* the present invention in teaching the use of a batch system that processes various sequence searches serially "one at a time" at a single site (the BCM Search Launcher server, see Abstract, lines 14-17 and page 461, column 2, discussing batch processing) instead of in parallel at multiple slave CPUs, as found in the present invention. Smith et al. is merely a client-server system for providing a search launcher WWW interface and merely provides access to existing WWW services on remote servers. No matter how the Office Action twists or mischaracterizes Smith et al. (i.e., "Smith et al. describes … promoting a distributed information space by filling out an HTML form…"), it is a fact that neither the client nor the BCM server include any step or software for splitting up a NxM dataset comparison into $n_N \times n_M$ tasks. Likewise, it is a fact that client search requests in Smith et al. are processed serially and that each search request is sent to a single remote site. A fair reading of Smith et al. illustrates that the disclosed system is merely a WWW gateway to pre-existing search services and that it can perform some pre-processing in the form of batch entry and post-processing in the form of adding links to results. It does nothing to solve the problems existing in the prior art, such as (1) that sequence-to-database comparisons (as illustrated in fig. 1 of Smith et al.) require large RAM requirements for efficient processing or (2) that typical BLAST queries over a network involve sending inefficient ASCII (256-bit)

characters (as illustrated by the "cut and paste" sequence entry disclosed by Smith et al.).

Likewise, Altschul et al., as a whole, *teaches away* from the present invention by teaching dataset-to-dataset comparison on a single machine (i.e., "a shared memory version of BLAST...loads the compressed DNA file into memory once" is the only disclosed technical performance enhancement). Although Altschul et al. discloses the comparison of two random sequences *n* and *m*, it nowhere suggests dividing the problem further, let alone dividing it into tasks for different computers to solve, as erroneously asserted by the Office Action.

The Office Action's citation to Reed et al. borders on the ridiculous. Reed et al. has nothing to do with bioinformatics. It has nothing to do with dataset comparisons. Indeed, it has nothing to do with solving large computational problems with distributed computing (but rather deals with information distribution). Reed et al. is drawn to an "automated communications system [that] operates to transfer data, metadata and methods from a provider computer to a consumer computer through a communications network." The disclosed compression in col. 57 is for word processing documents with PKZIP, not the databases of col. 14 as the Office Action implies. Like the other references, it *teaches away* from the present invention since, as the cited paper to Matsumoto et al. teaches on page 44, "if one applies the standard text compression software such as compress or gzip, they cannot compress DNA sequences, but only expand the file with more than two bits per symbol." The present invention applies standard redundancy reduction data compression to an

efficiently packed data element to avoid this issue, not the raw ASCII data that Smith et al. and Reed et al. seek to transmit over networks.

The stated motivation for the combination in the Office Action, i.e., that "it would have been obvious one having ordinary skill in the art at the time the invention was made to compress data (as stated by Altschul et al. and Reed et al.) and looping processes [sic] (as stated by Reed et al.) in order to offer enhanced, integrated, easy-to-use, and time-saving techniques to a large number of useful molecular biology database search and analysis services for organizing and improving access to these tools for Genome researchers worldwide (Smith et al., page 459, col 1, third paragraph to col. 2, first paragraph)" is not only incomprehensible, but it further is *completely unrelated to limitations of the claimed invention*. It is clearly an improper hindsight reconstruction, not even of the claimed invention, but merely for the purpose of combining the disparate references that the Examiner found that use appropriate words like "BLAST," "server," "network," "distributed," "database," and "compression," which apparently turned up in the required electronic text searches.

Indeed, the Office Action has completely failed at making a *prima facie* case of obviousness under *Graham v. Deere* since it has failed to identify or evaluate *any* of the differences between the claimed invention and the prior art.

No Reasonable Expectation of Success

One of ordinary skill in the art could not reasonably be expected to find Applicant's claimed invention for

comparing large datasets obvious in view of a plurality of references that provide no guidance on handling large datasets or processing them in parallel over a network. Indeed, if the compression teaching suggested by the Office Action were implemented (PKZIP compression of ASCII DNA data), the network would be saturated (and fail) due to the *expanded* file sizes that would result therefrom.

All Claim Limitations Not Shown

Smith et al. teaches the running of sequence-to-database searches, but fails to teach or fairly suggest numerous claim limitations required by all of the claims, including at least the following found in claims 1 and 13:

- dividing said query dataset N into n_N data elements having a size within a specified range [at client CPU];
- dividing said subject dataset M into n_M data elements having a size within said specified range[at client CPU];;
- determining a number of tasks for an entire comparison of datasets N and M as $n_N \times n_M$[at client CPU];;
- sending all data elements and task definitions to a master central processing unit (CPU) of a master-slave distributed computing platform,

  wherein task definitions comprise at least one comparison parameter, at least one executable element capable of performing comparisons, a query data element identification(ID)/descriptor, and a subject data element ID/descriptor, and

  wherein data elements are sent alternately from query and subject data elements;
- sending a task definition for each task from the master CPU to one of a plurality of slave CPUs when

all parts of a task definition and data elements referenced by said task definition are available at said master CPU;

- sending data elements referenced by said task definition to said slave CPU; and

- performing each task on a slave CPU.

Selection of a sequence to "clip and paste" into the HTML input form of Smith et al. is not a *division* of a query dataset N, but rather a specification of dataset N. No datasets in Smith et al. are ever divided, no tasks (*plural* for a single N-M comparison) are determined, and no subject dataset elements are ever sent to a Master CPU.

Altschul et al. fail to disclose any of the limitations missing from Smith et al. It merely discloses the basic BLAST algorithm for sequence comparison, i.e., comparing one sequence with another sequence, or for searching a database. Like Smith et al., Altschul et al. at least fail to disclose or suggest dividing sequence comparison problems into discrete segments for processing on a plurality of CPUs, let alone any specific method of doing this task.

Reed et al. also fail to remedy any of the defects of the Smith et al. and Altschul et al. references and is completely unrelated to the present invention.

As a whole, none of the cited prior art teaches or fairly suggests dividing the problem of comparing datasets M and N into $n_N \times n_M$ comparisons of data elements from N with data elements from M as presently claimed. For at least these reasons, Appellants submit that the claims are allowable over the prior art and request reconsideration and allowance of the claims.

Reply to the Examiner's Response to Arguments

While the Examiner acknowledges Appellants' arguments, the Final Office Action fails to adequately address many of them.

Grounds 1 and Grounds 2

In response to Appellants' alleged argument "that the term 'efficient' has an understood meaning in the art" (the actual arguments are that "efficient structure" in the context of the claim limitation "packing said data into an efficient structure" and "efficient encoding" are terms of art that are readily understood by one of ordinary skill in the art of bioinformatics) and that "the specification provides standards, in the form of examples" (the actual argument was that the specification included numerous examples, such as at paras. [19] and [68]), the Examiner only addressed the latter argument with the conclusory statement that "examples do not form an explicit definition including metes and bounds of this term." However, well known claim terms do not need explicit definitions. As stated in M.P.E.P. 2163, "The absence of definitions or details for well-established terms or procedures should not be the basis of a rejection under 35 U.S.C. 112" and "Information which is well known in the art need not be described in detail in the specification. See, e.g., *Hybritech, Inc. v. Monoclonal Antibodies, Inc.*, 802 F.2d 1367, 1379-80, 231 USPQ 81, 90 (Fed. Cir. 1986)."

Likewise, the Examiner fails to address the Appellants' arguments that the *Merriam-Webster* definition of "efficient" is inappropriate since known terms of art do

not need to be so defined. Indeed, the broadest reasonable interpretation of the claims must also be consistent with the interpretation that those skilled in the art would reach. *In re Cortright*, 165 F.3d 1353, 1359, 49 USPQ2d 1464, 1468 (Fed. Cir. 1999).

The Examiner takes the term "most efficient way" in Matsumoto et al. out of context to suggest it is a relative term. However, the phrase assumes true randomness and a binary storage means and was made in context of a further-compacting scheme based upon the fact that DNA molecules are not truly random. It *does* illustrate the well-established use of "efficient" in the field of bioinformatics.

Indeed, the only way that the Examiner can explain that the term "efficient" is indefinite (i.e., what characteristics must be present to contain the "desired effects"?) is to use a definition of the term outside the art of bioinformatics. The examples in the specification demonstrate that Appellant is using the definition of "efficient" that is well-known in the art of bioinformatics.

Grounds 3

The Examiner's response in the Final Office Action fails to address the Appellants' explanation for why the 1990 article by Altschul does not use "seed," "seed point," "sum," or "set." While the Examiner alleges that Appellants "attempt to define 'seed point and sum-set membership' by using various interpretations set forth in various articles" and "reference to a phrase in a published article does not automatically equate to it being well known in the art," Appellants submit that the cited publications are not

just any published articles, but are rather official BLAST documentation and peer-reviewed publications in scientific journals. If this evidence does not qualify as documentation for what was known to one of skill in the art, it is unclear what documentation beyond a technical treatise/dictionary the Examiner would consider sufficient.

Ground 5

With respect to the motivation or suggestion to combine references, the Examiner states that "Smith et al. state the problem of hindering efficient use as well as improving and simplifying access and sources which is a proper motivation to combine." Regardless, Smith et al. only suggests an improved interface with batch processing that *teaches against* the distributed processing of the present invention.

With regard to the claims not reciting the phrase "distributed computing," Appellants note that the claimed invention and the prior art must be considered *as a whole*. As a whole, the present claims define a distributed computing platform and method, whereas the cited prior art does not.

Likewise, the interpretation of the claimed "master-slave distributed computing platform" to be covered by the "system" of Smith et al. is clearly inconsistent with the specification, in contravention of M.P.E.P. 2111.

The Examiner's arguments related to "parallel use" not being recited in the claims again fails to address the claims as a whole, which claim dividing a comparison into tasks, sending the tasks to be computed on a plurality of CPUs, and returning task results - the very definition of parallel computing.

In regard to the Examiner's argument that not all limitations need to be found in each reference, Appellants note that (1) Smith et al. fails to teach or suggest that any dataset-to-dataset comparison is performed on more than one machine (the portal merely provides access to existing services), (2) Altschul et al. teaches dataset-to-dataset comparison on a single machine, and (3) Reed et al. has nothing to do with dataset comparisons.

With regard to the teachings of Reed et al., the Examiner misses the point that there is no reason to combine Reed et al. with the bioinformatics references absent impermissible hindsight. Smith et al. teaches basic clip-n-paste submissions of queries to run against external databases. Ordinary results are returned. There is no suggestion of compression and there is no metadata, so there is no reason to strip it. Likewise, Altschul et al. teaches a basic sequence-to-sequence comparison on a local machine. Again, no metadata or compression is needed or desirable. The only reason to look to the diverse art of Reed et al. is the Appellants' disclosure.

With regard to Smith et al. on pages 15-16, the quoted portions of the Office Action merely repeat the unsupported contentions in a long narrative that fails to match claim limitations with specific portions of the prior art. For example, if the Examiner were to try a proper comparison between the claims and the prior art under Graham v. Deere, he would immediately see that a client CPU in Smith et al., consisting of a computer with a browser, fails to include instructions for dividing datasets N and M, as required.

CONCLUSION

For the above reasons, Appellants respectfully submit that the present claims meet the requirements of 35 U.S.C. 112 and that the Examiner has failed to make out a *prima facie* case of obviousness under 35 U.S.C. 103 with regard to claims 1, 4, 6-13, and 18-23 and asks that the obviousness rejection be reversed.

Respectfully submitted,

Christopher B. Kilner
Registration No. 45,381
Roberts Abokhair & Mardula, LLC
11800 Sunrise Valley Drive,
Suite 1000
Reston, VA 20191
(703) 391-2900

VIII - Appendix of Claims on Appeal

1. A method of comparing a query dataset N with a subject dataset M, comprising:

dividing said query dataset N into n_N data elements having a size within a specified range;

dividing said subject dataset M into n_M data elements having a size within said specified range;

determining a number of tasks for an entire comparison of datasets N and M as $n_{N \times} n_M$;

sending all data elements and task definitions to a master central processing unit (CPU) of a master-slave distributed computing platform,

wherein task definitions comprise at least one comparison parameter, at least one executable element capable of performing comparisons, a query data element identification(ID)/descriptor, and a subject data element ID/descriptor, and

wherein data elements are sent alternately from query and subject data elements;

sending a task definition for each task from the master CPU to one of a plurality of slave CPUs when all parts of a task definition and data elements referenced by said task definition are available at said master CPU;

sending data elements referenced by said task definition to said slave CPU;

performing each task on a slave CPU; and

returning task results for each task to said master CPU.

2. The method of claim 1, further comprising randomizing sequence order of each dataset if either dataset contains related sequences in a contiguous arrangement.

3. The method of claim 1, further comprising formatting said datasets so as to use exactly the same ambiguity substitutions.

4. The method of claim 1 wherein dividing said datasets into data elements further comprises:

stripping all metadata from data;

packing said data into an efficient structure;

creating an index for said data and packing said index and said data in an uncompressed data structure; and

compressing said uncompressed data structure into a data element using a redundancy reduction data compression method.

5. The method of claim 1, further comprising sending remaining data elements from a more numerous of said datasets to said master CPU followed by all task definitions for otherwise complete tasks if there are fewer data elements from one dataset.

6. The method of claim 1 wherein performing a task on said slave CPU further comprises:

uncompressing and unpacking data from said query and subject data elements;

looping through query sequences from said query data element to perform setup, preprocessing and table generation for each row of comparisons;

looping through subject sequences from said subject data element and, for each pair of query and subject sequences, performing a comparison using said executable element and finding results based on said at least one comparison parameter; and

storing minimal information that will allow reconstruction of said result.

7. The method of claim 6 wherein storing said minimal information comprises:

storing index information for said query and said subject sequence;

storing bounds information for start and stop of said query and subject sub sequences;

storing data that quantify fulfillment of significance criteria for a significant match; and

storing an efficiently encoded representation of alignment between said bounds corresponding to a high-scoring segment pair.

8. The method of claim 7, further comprising storing a seed point and sum-set membership for each alignment for Basic Local Alignment Search Tool (BLAST).

9. The method of claim 7, further comprising storing task results in a task result file, said file including query and subject sequence data and metadata corresponding to the task that the results came from, metadata for the subject sequence, the partial subject sequence data corresponding to the subject bounds of the significant alignment result, and any other results data for each result in the task results.

10. The method of claim 9, further comprising generating a BLAST report for each query data element.

11. The method of claim 10, further comprising concatenating results from all BLAST reports to produce a text file identical to a blastall run of said query and subject datasets.

12. The method of claim 1 wherein said datasets are selected from the group consisting of genomic and proteomic databases.

13. A system for comparing a query dataset N with a subject dataset M, comprising:

a master central processing unit (CPU) of a master-slave distributed computing platform;

a plurality of slave CPUs capable of communication with said master CPU; and

a client CPU with instructions for:

dividing said query dataset N into n_N data elements having a size within a specified range;

dividing said subject dataset M into n_M data elements having a size within said specified range;

determining a number of tasks for an entire comparison of datasets N and M as $n_{N \times} n_M$;

sending all data elements and task definitions to said master CPU of a master-slave distributed computing platform,

wherein task definitions comprise at least one comparison parameter, at least one executable element capable of performing comparisons, a query data element identification (ID)/descriptor, and a subject data element ID/descriptor, and

wherein data elements are sent alternately from query and subject data elements;

said master CPU comprising instructions for:

sending a task definition for each task to one of said plurality of slave CPUs when all parts of a task definition and data elements referenced by said task definition are available at said master CPU; and

sending data elements referenced by said task definition to said slave

CPU; and

said slave CPUs including instructions for:

performing each task; and

returning task results for each task to said master CPU.

14. The system of claim 13, further comprising means for randomizing sequence order of each dataset if either dataset contains related sequences in a contiguous arrangement.

15. The system of claim 13, further comprising means for formatting said datasets so as to use exactly the same ambiguity substitutions.

16. The system of claim 13, wherein said instructions for dividing said datasets into data elements further comprises instructions for:

stripping all metadata from data;

packing said data into an efficient structure;

creating an index for said data and packing said index and said data in an uncompressed data structure; and

compressing said uncompressed data structure into a data element using a redundancy reduction data compression method.

17. The system of claim 13, further comprising instructions for sending remaining data elements from a more numerous of said datasets to said master CPU followed by all task definitions for otherwise complete tasks if there are fewer data elements from one dataset.

18. The system of claim 13, wherein instructions for performing a task on said slave CPU further comprises instructions for:

uncompressing and unpacking data from said query and subject data elements;

looping through query sequences from said query data element to perform setup, preprocessing and table generation for each row of comparisons;

looping through subject sequences from said subject data element and, for each pair of query and subject sequences, performing a comparison using said executable element and finding results based on said at least one comparison parameter; and

storing minimal information that will allow reconstruction of said result.

19. The system of claim 18, wherein said instructions for storing said minimal information comprises instructions for:

storing index information for said query and said subject sequence;

storing bounds information for start and stop of said query and subject sub sequences;

storing data that quantify fulfillment of significance criteria for a significant match; and

storing an efficiently encoded representation of alignment between said bounds corresponding to a high-scoring segment pair.

20. The system of claim 19, further comprising instructions for storing task results in a task result file, said file including query and subject sequence data and metadata corresponding to the task that the results came from, metadata for the subject sequence, the partial subject sequence data corresponding to the subject bounds of the significant alignment result, and any other results data for each result in the task results.

21. The system of claim 20, further comprising instructions for generating a Basic Local Alignment Search Tool (BLAST) report for each query data element.

22. The system of claim 21, further comprising means for concatenating results from all BLAST reports to produce a text file identical to a blastall run of said query and subject datasets.

23. The system of claim 13, wherein said datasets are selected from the group consisting of genomic and proteomic databases.

IX. EVIDENCE APPENDIX

Matsumoto et al.

Altschul et al.

Varre et al.

Karlin et al.

BLAST Help Manual

BLAST 2.0 Release Notes

Anderson et al.

Brudno et al.

Genome Informatics 11: 43-52 (2000)

Biological Sequence Compression Algorithms

Toshiko Matsumoto[1,3] toshikom@is.s.u-tokyo.ac.jp
Kunihiko Sadakane[2] sada@dais.is.tohoku.ac.jp
Hiroshi Imai[1] imai@is.s.u-tokyo.ac.jp

[1] Department of Information Science, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
[2] Graduate School of Information Sciences, Tohoku University, 2-1-1 Katahira, Aoba-ku, Sendai 980-8577, Japan
[3] Presently, the author is with Hitachi Software Engineering Co., Ltd.

Abstract

Today, more and more DNA sequences are becoming available. The information about DNA sequences are stored in molecular biology databases. The size and importance of these databases will be bigger and bigger in the future, therefore this information must be stored or communicated efficiently. Furthermore, sequence compression can be used to define similarities between biological sequences.

The standard compression algorithms such as `gzip` or `compress` cannot compress DNA sequences, but only expand them in size. On the other hand, *CTW* (Context Tree Weighting Method) can compress DNA sequences less than two bits per symbol. These algorithms do not use special structures of biological sequences.

Two characteristic structures of DNA sequences are known. One is called palindromes or reverse complements and the other structure is approximate repeats. Several specific algorithms for DNA sequences that use these structures can compress them less than two bits per symbol.

In this paper, we improve the *CTW* so that characteristic structures of DNA sequences are available. Before encoding the next symbol, the algorithm searches an approximate repeat and palindrome using hash and dynamic programming. If there is a palindrome or an approximate repeat with enough length then our algorithm represents it with length and distance. By using this preprocessing, a new program achieves a little higher compression ratio than that of existing DNA-oriented compression algorithms. We also describe new compression algorithm for protein sequences.

Keywords: DNA, protein, compression, context tree weighting

1 Introduction

Today, the complete DNA sequences of many organisms are already known, and the completion of human genome project is making steady progress. The information of DNA sequences, RNA sequences, and amino-acid sequences of proteins are stored in molecular biology databases. It is well known that the sizes of these databases are increasing nowadays very fast. Therefore it is needed to store and communicate data efficiently. Furthermore, there are other reasons to study compression of biological sequences.

Understanding the Properties of DNA Sequences Using Compression Algorithms Since DNA sequences contain four symbols 'a,' 't,' 'g,' and 'c,' if these were totally random, the most efficient way to represent them would be using two bits for each symbol. However, only a small fraction of DNA sequences result in a viable organism, therefore the sequences which appear in a living organism are expected to be nonrandom and have some constraints. In other words, they should be compressible. The studies in compression algorithms for DNA sequences answer the basic

question about the compressibility of DNA sequences, and from a viewpoint of information science, we can use compression techniques to capture the properties of DNA sequences. It is known that DNA sequences have two characteristic structures. One is reverse complements, and the other is approximate repeats. The reverse complement of a sequence is a reverse sequence whose each symbol is replaced with its complement one. The approximate repeats are repeats that contains errors. There have been developed several special-purpose compression algorithms for DNA sequences (Grumbach and Tahi [3], Chen, Kwong and Li [1], Lanctot, Li and Yang [5]). These algorithms use the structures and can achieve high compression ratio.

There is a difference between the compression ratio of coding and non-coding regions of DNA sequences, and this is supported by a biological hypothesis (Lanctot, Li and Yang [5]). Not all of the DNA sequence specify a protein. In higher eukaryotes (such as plants and animals) much of the sequence is cut out before the cell translates it into protein. Random mutations in a DNA sequence are thought to be more deleterious if they take place in a coding regions rather than in a non-coding regions. Therefore the two regions should have different information theoretic entropy.

With conditional compression ratio, we can evaluate the "distance" between pairs of DNA sequences (Chen, Kwong and Li [1]). DNA sequences that are "close" to each other are required to be "close" to each other on the evolutionary tree, thus the "distance" on the evolutionary tree can be measured by compression algorithms. Therefore we can guess whether organisms are "close" on the evolutionary tree using compression algorithms. The minimum alignment score also can be used to estimate the distance between a pair of sequences, however it is good only to measure two genes that are closely related. It can hardly handle simple changes like reversal, translocation, and shuffling. Using conditional compression ratio is more robust than using the minimum alignment score. Note that we can choose a compression algorithm for defining similarities of sequences so that the compression ratio and the score of the alignment have one-to-one correspondence. Thus compression of DNA sequence is important not only for improvement of efficiency of storage or communication but also for understanding the properties of DNA sequences.

Using DNA Sequences as a Challenging Subject for Compression Algorithms DNA sequences only contain four symbols, therefore two bits per symbol is enough to represent these sequences even if they are totally random. However if one applies the standard text compression software such as `compress` or `gzip`, they cannot compress DNA sequences but only expand the file with more than two bits per symbol. Thus DNA sequences are important as a new challenge for study of compression algorithms. There are some reasons pointed out. These software are designed mainly for English text compression, while the regularities in DNA sequences are much subtler (Chen, Kwong and Li [1]). Generally the windows of the methods based on dictionary have a fixed width of small size. The use of small windows is efficient on text whose redundancy is local. However, in the case of DNA sequence, redundancies may occur at very long distances and factors can be very long (Grumbach and Tahi[3]).

Huffman's code also fails badly on DNA sequences both in the static and adaptive model, because there are only four kind symbols in DNA sequences and the probabilities of occurrence of the symbols are not very different.

Concerning compression ratio, *PPM* (Cleary and Witten[2]) is one of the best compression algorithms in practice. However it cannot compress DNA sequences less than two bits per symbol either.

It is true that the compression of DNA sequence is a difficult task for general compression algorithms, but at the same time, from the viewpoint of compression theory it is an interesting subject for understanding the properties of various compression algorithms.

Contributions of this paper Reverse complements and approximate repeats are known as characteristic structures of DNA sequences. We introduce a new DNA-oriented compression algorithm that uses context tree weighting and takes account of the characteristic structures of DNA sequences.

The new algorithm has a function for searching reverse complements or approximate repeats using hash table and dynamic programming, and if there is one, the algorithm represents it by its length and distance. Our new algorithm can achieve a little higher compression ratio than that of existing special purpose compression algorithms for DNA sequences.

It is known that compression of proteins is a difficult task (Nevill-Manning and Witten [7]). The standard compression algorithms such as `gzip` or `compress` cannot compress proteins but only expand them in size. There is a special purpose compression algorithm for proteins that takes account of the underlying biochemical principles and it can compress proteins, although the compression ratio is not very high. Therefore proteins are said to be incompressible. We show that many general compression algorithms can really compress proteins.

2 Compression Algorithms

We briefly explain two compression algorithms used in our algorithm.

2.1 PPM

PPM (Cleary and Witten [2]) is a kind of statistical compression algorithms and has a high compression ratio. *PPM* predicts the probability of next symbol using preceding several symbols called context, and then compresses a sequence of symbols one by one with this probability. The maximum value of length d of the context is called *order* of *PPM*. This value is one of parameters of *PPM*.

Calculate Probability using Context For each substring of input data whose length is less than or equal to *order*, *PPM* stores the frequency of the each symbol that appears after the context. With this values, *PPM* estimates the probability of the next symbol.

Escape Symbol A special escape symbol *esc* is defined in the *PPM* algorithm for an appearance of a novel symbol which cannot be expected from the information of frequencies for the context whose length is d. The *esc* symbol is a special symbol for shortening the length of the context. If the decoder find the symbol *esc*, it changes the context to one whose length is $d-1$. For an appearance of a novel symbol which has never appeared and *PPM* has no information of frequency about the symbol, the context whose length is -1 is defined null context. In the context, all possibilities of symbols which can appear are equal.

The Value of *order* If the value of *order* becomes bigger, the precision of the prediction is improved. However on the other hand, the flexibility of the prediction is lost and the frequency of *esc* increases. The increase of the frequency of *esc* has a bad influence on the compression ratio, therefore there is an optimal value of *order*. For each sequence, the optimal value of *order* exists. It is well known that the optimal value of *order* is five for many English texts. For DNA sequences, in many cases the optimal value of *order* is less than five.

2.2 Context Tree Weighting

Context Tree Weighting (*CTW*) is a universal compression algorithm for FSMX sources proposed by Willems et al [11], and expected to have good compression ratio with an unknown model and unknown parameters. FSMX sources are related to Tree sources with the property that the next symbol probabilities depend on preceding several symbols. The PPM algorithm uses only one model, but the *CTW* guesses the probabilities by adding up all possible models using weighting.

Context Tree The contexts are represented in a tree form and called a context tree. Each node of the tree represents a context. All the tree have to satisfy is the restrictions of FSMX sources. for convenience of explanation we assume that the maximum depth of the context tree is D. At each node one store the frequency of the each symbol that appears after the context. Each value of the frequencies of a parent node is the summation of the values of its children.

Probabilities at Each Context For each context, the probability of the next symbol is estimated with the frequencies of symbols stored in the corresponding node of the context tree. For the probability of a symbol c at a context s we write $P_e^s(c)$. This value is calculated by the concept of the PPM algorithm. In the original algorithm of Willems et al [11] the Krichevski-Trofimov (KT) probability estimator is used. In the PPM algorithm, a special escape symbol esc is used. If a novel symbol c appears in a context s_d which has depth of d, esc is encoded in context s_d and then c is encoded in a context s_{d-1} which has depth of $d-1$. To use the idea of esc in the CTW program, the estimate for the probability of the symbol c is the probability of esc in the context s_d times the probability of c in the shorter context s_{d-1}. In the null context λ, probabilities of symbols they have not appeared are all equal and they divide the escape probability equally among themselves. We denote by $\mathcal{A}$ the set of all alphabets. The probability $P_e^{s_d}(c)$ is calculated as follows:

1. calculate $P_e^{\lambda}(c)$
   (a) let $m = 0$
   (b) for all $c \in \mathcal{A}$ if c has not appeared, $m = m + 1$
   (c) for all $c \in \mathcal{A}$
   if c has appeared, $P_e^{\lambda}(c)$ is calculated according as PPM
   else, $P_e^{\lambda}(c) = P_e^{\lambda}(esc)/m$
2. calculate $P_e^{s_d}$ $(1 \le d \le D)$
   (a) let $e = 0$
   (b) for all $c \in \mathcal{A}$ if c has not appeared in the context s_d, $e = e + P_e^{s_d}(c)$
   (c) for all $c \in \mathcal{A}$
   if c has appeared in the context s_d, $P_e^{s_{d-1}}(c)$ is calculated according as PPM
   else, $P_e^{s_{d-1}}(c) = P_e^{s_{d-1}}(esc) \cdot P_e^{s_d}(c)/e$

Adding up Models Assume that a symbol x_t has a context $0s$. For each context s, according the following expression P_w^s is calculated. P_w^s is the weighted probability under P_e^s, and the next symbol is encoded on the basis of this value at the null context P_w^{λ}. γ is a weighting parameter of CTW and determine the importance of long or short contexts. If γ is large, CTW regards the short contexts as important, and if γ is small, CTW regards the long contexts as important. Note that $0 \le \gamma \le 1$.

$$P_e^s(x_1^t) = \prod_{1 \le i \le t \text{ and the context of } x_i \text{ is } s} P_e^s(x_i)$$

$$P_w^s(x_1^t) = \begin{cases} \gamma P_e^s(x_1^t) + (1-\gamma) \prod_{c \in \mathcal{A}} P_w^{cs}(x_1^t) & \text{(node } s \text{ is not a leaf)} \\ P_e^s(x_1^t) & \text{(node } s \text{ is a leaf)} \end{cases}$$

$$\begin{aligned} P_w^s(x_t) &= \frac{P_w^s(x_1^t)}{P_w^s(x_1^{t-1})} = \frac{\gamma P_e^s(x_1^t) + (1-\gamma) \prod_{c \in \mathcal{A}} P_w^{cs}(x_1^t)}{\gamma P_e^s(x_1^{t-1}) + (1-\gamma) \prod_{c \in \mathcal{A}} P_w^{cs}(x_1^{t-1})} \\ &= \frac{\gamma P_e^s(x_1^{t-1}) P_e^s(x_t) + (1-\gamma) P_w^{0s}(x_t) \prod_{c \in \mathcal{A}} P_w^{cs}(x_1^{t-1})}{\gamma P_e^s(x_1^{t-1}) + (1-\gamma) \prod_{c \in \mathcal{A}} P_w^{cs}(x_1^{t-1})} \end{aligned}$$

By defining

$$\beta_s(x_1^{t-1}) = \frac{P_e^s(x_1^{t-1})}{\prod_{a \in A} P_w^{es}(x_1^{t-1})},$$

we obtain the expression

$$\begin{aligned} P_w^s(x_t) &= \frac{\gamma\beta P_e^s(x_t) + (1-\gamma)P_w^{0s}(x_t)}{\gamma\beta + (1-\gamma)} \\ &= \frac{\gamma\beta}{\gamma\beta + (1-\gamma)} P_e^s(x_t) + \frac{1-\gamma}{\gamma\beta + (1-\gamma)} P_w^{0s}(x_t) \end{aligned}$$

The initial value of β is 1.0 because if x_1^{t-1} is a null sequence then $P_e^s(x_1^{t-1})$ is 1.0 and thus $P_w^s(x_1^{t-1})$ is 1.0. Therefore β can be computed incrementally as follows:

$$\beta_s(x_1^t) = \frac{P_e^s(x_1^t)}{\prod_{a \in A} P_w^{es}(x_1^t)} = \beta_s(x_1^{t-1}) \cdot \frac{P_e^s(x_t)}{P_w^{0s}(x_t)}$$

3 DNA-Oriented Compression Algorithms

It is known that DNA sequences have characteristic structures that cannot be observed in other kinds of data such as English text. There are several special purpose compression and entropy estimating algorithms for DNA sequence that use these structures are studied and the compression ratio of these algorithms are better than two bits per symbol (Grumbach and Tahi [3], Chen, Kwong and Li [1], Lanctot, Li and Yang [5]).

3.1 Characteristic Structures of DNA Sequences

Reverse Complement In DNA sequences, the symbols 'a' and 't' are the complement of each other, and the symbols 'g' and 'c' are also the complement to each other. A string y_1^n is the reverse complement of x_1^n if x_i and y_{n+1-i} are the complement of each other for $1 \leq i \leq n$, and a pair of strings y_1^n and x_1^n is called palindrome. For example, the reverse complement of *aaacgt* is *acgttt*.

There are some DNA sequences which have long reverse complements. "CHMPXX" is the complete chromosome III from yeast and one of the standard benchmark sequences used in DNA-oriented compression and entropy estimating algorithms. The benchmark sequences are available at [6]. "CHMPXX" has 121024 symbols in it and it has an about 10000 symbols long reverse complement. "VACCG" is the complete gene of a virus and also one of the standard benchmark sequence. "VACCG" has 191737 symbols in it and it has an about 8000 symbols long reverse complement. Therefore the specific redundancy is important for compression algorithms.

Approximate Repeats DNA sequences, especially ones of higher eukaryotes, have many repeats. It has been conjectured that this is because genes duplicate themselves sometimes for evolutionary or simply for "selfish" purposes. Containing many repeats is favorable for compression algorithms, however such regularities are often blurred by random mutations. Therefore it is important to adapt to repeats that contain errors.

3.2 DNA-oriented Compression Algorithms

Biocompress-2 Grumbach and Tahi [3] proposed lossless compression algorithms for DNA sequence, namely Biocompress-2. The algorithm is based on LZ77. Biocompress-2 searches for exact repeats or reverse complements and encodes them with length and position of it. Literal encoding and second order arithmetic encoding is also used. In literal encoding each symbol is encoded as a two bit number. Biocompress-2 compares these three methods and chooses the most efficient one dynamically.

GenCompress Chen, Kwong and Li [1] developed GenCompress that is also a compression algorithm for DNA sequence based on LZ77. GenCompress uses both approximate repeats and reverse complements and also uses reverse complements that contain errors. The algorithm searches approximate repeats or approximate reverse complements, and encodes it with length, position and the errors. If an approximate repeat or an approximate reverse complement contains many errors, it does not provide profit in the encoding, therefore GenCompress uses second order arithmetic encoding.

4 New DNA-Oriented Compression Algorithms Using Context Tree Weighting

We propose a new DNA compression algorithm. It is a combination of LZ77-type algorithm like *GenCompress* and the CTW algorithm. Long exact/approximate repeats are encoded by LZ77-type algorithm, while short repeats are encoded by the CTW. We also use heuristics to improve compression ratios described as follows.

4.1 Judgment of Using Edit Operations

Our new function searches approximate repeats or approximate reverse complements using dynamic programming. With more edit operations the length can be enlarged, however we must determine where to stop dynamic programming to maximize the profit and improve the compression ratio. The algorithm estimates the number of bits needed to encode the repeat by *CTW* using the compression ratio of the sequence already encoded, and find the combination of edit operations that provides the biggest difference. When the length is short, the distance is long or many edit operations is needed, the structure cannot provide profit and then the algorithm does not use it.

4.2 Non-Greedy Search of Repeats

If the algorithm searches reverse complements or repeats greedy, it may miss longer one. This is illustrated in Figure 1. To cope with this, we defer the selection of these structures with a lazy evaluation mechanism (Horspool [4]). After a reverse complement or an approximate repeat of length l has been found, the algorithm searches for a longer structure at the next M symbols. If another reverse complement or approximate repeat is found and that provides more profit, the previous one is abandoned. Otherwise, the original one is kept.

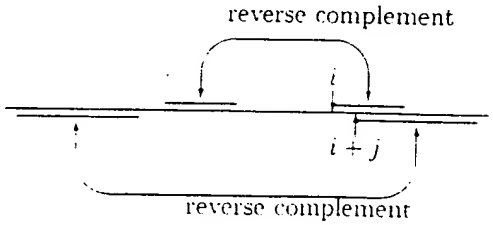


Figure 1: Overlapping two reverse complements.

1. find an optimal reverse complement or repeat for v_i^2 which begins from the current position , the length of the structure is denoted l_0.
2. if l_0 is smaller than lb, goto 9.

3. calculate the number of bits needed to encode l_0 symbols using LZ77-like function and store in LZ_0.
4. estimate the number of bits needed to encode l_0 symbols using CTW and store in CTW_0.
5. if $LZ_0 < CTW_0$ (the structure does not provide profit) goto 9.
6. for $k = 1$ to M
   (a) find an optimal reverse complement or repeat for v_{l+k}^{∞}, the length of the structure is denoted by l_k.
   (b) if l_k is larger than l_0, do the following.
      i. calculate the number of bits needed to encode l_k symbols using LZ77-like function and store in LZ_k.
      ii. estimate the number of bits needed to encode l_k symbols using CTW and store in CTW_k.
7. if $LZ_0 - CTW_0$ is larger than $LZ_k - CTW_k$ for $1 \leq k \leq M$ then encode the structure using LZ77-like function and goto 1.
8. find k such that $LZ_k - CTW_k$ is larger than $LZ_{k'} - CTW_{k'}$ for $0 \leq k' \leq M$ and encode $k - 1$ symbols using CTW and goto 1.
9. encode one symbol using CTW and goto 1.

4.3 Experimental Results

Environment of Experiments We use a SUN Ultra60 workstation (CPU Ultra SPARC-II 360 MHz, memory 2048MB) and a Sun Enterprize 450 workstation (CPU Ultra SPARC-II 4x400MHz, memory 4096MB) running Solaris 2.7. If the algorithm searches reverse complements or approximate repeats, more time is needed to execute program than original CTW. And the speed of non-greedy algorithm is slower than that of greedy-algorithm. The maximum number of edit operations also effects the speed. If the sequence is long or contains many reverse complements or approximate repeats, much time is needed. For short sequences such as "HUMDYSTROP" we need about 8 minutes and for long sequences such as "HEHCMVCG" or "SCCHRIII" we need some hours. In many cases the optimal value of *order* of CTW is 32 (Sadakane, Okazaki, Matsumoto and Imai [9]), therefore we use this value. For the value of γ which indicates the importance of long and short contexts, we examined various values and checked the effect and tendency of γ. The non-greedy algorithm checks the next 32 symbols.

Compression Ratio of Each Algorithm The compression ratio of each algorithm is shown in Table 1. *Biocompress-2* (Grumbach and Tahi [3]) and *GenCompress* (Chen, Kwong and Li [1]) are DNA oriented compression programs. When our algorithm can achieve higher compression ratio than *Biocompress-2* and *GenCompress*, the compression ratio are written in bold face.

normal CTW The original CTW program.

CTW+LZ A non-greedy program which searches exact and approximate reverse complements and exact and approximate repeats. This program encodes these structures using LZ77-like function and edit operations are encoded by arithmetic coding. Symbols which are not encoded in a repeat are encoded by order-32 CTW.

In the most cases, our new program can achieve a little higher compression ratio than *Biocompress* and *GenCompress*.

Table 1: Compression ratios of algorithms which encode repeats by using LZ77-like function.

Sequence name	Sequence length	*normal CTW*	*CTW+LZ*	*Biocompress-2*	*GenCompress*
CHMPXX	121024	1.8381	**1.6690**	1.6848	1.673
CHNTXX	155844	1.9330	**1.6129**	1.6172	1.6146
HEHCMVCG	229354	1.9584	**1.8414**	1.848	1.847
HUMDYSTROP	38770	**1.9200**	**1.9175**	1.9262	1.9226
HUMGHCSA	66495	1.3638	**1.0972**	1.3074	1.1048
HUMHBB	73308	1.8928	**1.8082**	1.88	1.8204
HUMHDABCD	58864	1.8973	1.8218	1.877	**1.8192**
HUMHPRTB	56737	1.9132	**1.8433**	1.9066	1.8466
MPOMTCG	186609	1.9624	**1.9000**	1.9378	1.9058
PANMTPACGA	100314	1.8664	**1.8555**	1.8752	1.8624
VACCG	191737	1.8577	1.7616	**1.7614**	**1.7614**
average	–	1.8547	**1.7389**	1.7837	1.7434

5 Protein Compression Algorithms

Proteins are sequences drawn from amino acids. There are 20 kinds of amino acids except for some peculiar ones, therefore the size of alphabet of proteins is 20. It is known that the compression of proteins is also very difficult (Nevill-Manning and Witten [7]). The size of alphabet is 20, consequently the entropy of proteins is equal to or less than $\log_2(20) = 4.322$ per symbol. However the compression ratios by the widely used compression algorithms `compress` or `gzip` are more than $\log_2(20)$ bits per symbol. Though *PPMD+* can achieve high compression ratio for English text, it cannot compress proteins either.

Compression results are shown in Table 2. The unit of compression ratio is bit per symbol. The proteins are used in a protein-oriented compression algorithm (Nevill-Manning and Witten [7]) and available at [8]. When an algorithm can compress a sequence less than $\log_2(20)$ bits per symbol, the corresponding compression ratio is written in bold face.

compress, *bzip2* and *gzip* are widely used compression programs `compress`, `bzip2` and `gzip`. *normal PPMD+* gives the results for the statistical compression program *PPMD+* (Teahan [10]). The value of *order* is set to 5 which is known as the best value for compression ratio of English text. *adapted PPMD+* is a modified *PPMD+* program whose value of *order* is adapted. We test the value of *order* from 0 to 15 and the optimal *order* for each sequence is in parenthesis next to compression ratio.

arith implements the adaptive arithmetic coding. *lz-ari* is the enhanced arithmetic coding with an LZ77-like function. The size of alphabet is 20.

The values of *CTW20* are results of an improved *CTW* program whose size of alphabet is changed to 20. The value of *order* is represented in parenthesis. We examine the effect of the value of γ. In many case about 0.005 is the optimal, and in Table 2, the best values are given. We cannot examine the compression ratio of *CTW20(16)* for Human and Saccharomyces Cerevisciae due to lack of memory. *lz-CTW* encodes exact repeats. Single symbols are encoded by order-8 CTW. *lza-CTW* is the same as the *lz-CTW*, except that it encodes approximate repeats by an LZ77-like function.

CP (Nevill-Manning and Witten [7]) is a protein-oriented compression algorithm on the basis of *PPM* and takes account of the underlying biochemical principles. The algorithm uses the probabilities that an amino acid will mutate to another and weights the contexts.

As widely used `compress` and `gzip`, in all cases cannot compress proteins less than $\log_2(20)$ bits per symbol. They only expand in size. `bzip2` can compress three proteins less than $\log_2(20)$ bits per

Table 2: The results of general compression algorithms and proteins.

Sequence name	Haemophilus Influenzae	Human	Methanococcus Janaschii	Saccharomyces Cerevisciae
Sequence length	509519	3295751	448779	2900352
$\log_2 20$	4.322	4.322	4.322	4.322
compress	4.7702	4.7177	4.6459	4.7761
bzip2	4.324	**4.256**	**4.269**	**4.300**
gzip -9	4.6712	4.6054	4.5879	4.6397
arith	**4.1557**	**4.1328**	**4.0679**	**4.1627**
lz-ari(1M)	**4.1270**	**4.0258**	**4.0445**	**3.9973**
normal PPMD+	4.862	4.641	4.711	4.686
adapted PPMD+	**4.151(1)**	**4.119(0)**	**4.061(1)**	**4.157(1)**
CTW20(8)	**4.1381**	**4.0368**	**4.0513**	**4.0293**
CTW20(16)	**4.1378**	—	**4.0510**	—
lz-CTW(8)	**4.1185**	**4.0055**	**4.0323**	**3.9870**
lza-CTW(8)	**4.1177**	**3.9203**	**4.0279**	**3.9514**
CP(1)	**4.149**	**4.158**	**4.060**	**4.126**
CP(2)	**4.146**	**4.152**	**4.056**	**4.120**
CP(3)	**4.143**	**4.112**	**4.051**	**4.146**

symbol.

arith can compress all proteins less than $\log_2(20)$ bits per symbol. *lz-ari* also can compress all proteins. Although in all cases *normal PPMD+* only expands in size, *adapted PPMD+* can really compress all of the proteins less than $\log_2(20)$ bits per symbol.

CTW20 can compress each protein and the results of *CTW20(16)* are higher a little than *CTW20(8)*. The optimal values of γ are 0.0005, 0.0005, 0.001 and 0.0005 for Haemophilus Influenzae, Human, Methanococcus Janaschii and Saccharomyces Cerevisciae. The variation of the compression ratio by changing the value of γ is small, just like the case of original *CTW*.

lz-CTW also can compress all proteins. The difference between the compression ratio of *lz-ari* and *lz-CTW* indicates the difference between the power of arithmetic coding and *CTW*. The compression ratio of *CTW* is improved because of LZ77-like function, therefore the sequences have repeats in it. Furthermore, the compression ratio, especially for Human, is improved by the *lza-CTW* that encodes approximate repeats. Each sequence is a connection of proteins of a organism, therefore the LZ77-like function can find repeat both from the same protein and from the previous proteins. The existence of exact and approximate repeats in a sequence may indicate that proteins have repeats, and may indicate that a organism has many similar proteins. Note that it is possible that both are true. We have no idea.

CP can compress all of the proteins less than $\log_2(20)$ bits per symbol. The compression ratio are improved by enlarging the value of *order*, however the gains are little. This appears that little compression is possible using algorithms that rely on Markov dependence (Nevill-Manning and Witten [7]). However, this algorithm will be improved by using our techniques to combine statistical compression algorithms with CTW.

If there are some characteristic structures in proteins, the special purpose algorithms that use the structures can achieve high compression ratio.

6 Concluding Remarks

We have proposed DNA and protein sequence compression algorithms. For DNA sequences, our algorithm slightly outperforms *GenCompress* by encoding bases which are not encoded by repeats by *CTW*. For protein sequences, our algorithms significantly improves the result of the protein-oriented compression algorithm. Furthermore, ours will be improved by combining *CTW* with the protein-oriented algorithm.

Though improvements of our algorithms seem to be small, the improvements are achieved for most of the examined sequences. Therefore our algorithms can be used to define more precise similarities between sequences. This is important to classify biological sequences and make phylogeny trees.

Acknowledgement

The work of the second author was supported in part by the Grant-in-Aid on Priority Areas (C), 'Genome Information Science,' of the Ministry of Education, Science, Sports and Culture of Japan. The work of the third author was supported in part by the Grant-in-Aid on Priority Areas (A), 'Genome Science.'

References

[1] Chen, X., Kwong, S., and Li, M., A compression algorithm for DNA sequences and its applications in genome comparison. *Genome Informatics*, 10:52–61, December 1999.

[2] Cleary, J.G. and Witten, I.H., Data compression using adaptive coding and partial string matching. *IEEE Trans. on Commun.*, COM-32(4):396–402, April 1984.

[3] Grumbach, S. and Tahi, F., A new challenge for compression algorithms: genetic sequences, *Information Processing & Management*, 30:875–886, 1994.

[4] Horspool, R.N., The effect of non-greedy parsing in Ziv-Lempel compression methods, *Proc. of IEEE Data Compression Conference*, 302–311, 1995.

[5] Lanctot, J.K., Li, M., and Yang, E., Estimating DNA sequence entropy. *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, 409–418, 2000.

[6] National Center for Biotechnology Information, Entrez Nucleotide Query, `http://www.ncbi.nlm.nih.gov/htbin-post/Entrez/query?db=n_s`.

[7] Nevill-Manning, C.G. and Witten, I.H., Protein is incompressible, *Proc. of IEEE Data Compression Conference*, 257–266, 1999.

[8] Craig G. Nevill-Manning. Protein is incompressible, `http://sequence.rutgers.edu/DCC99/`.

[9] Sadakane, K., Okazaki, T., Matsumoto, T., and Imai, H., Implementing the context tree weighting method by using conditional probabilities. *Proc. of 22th Symposium on Information Theory and its Applications*, 673–676, SITA, December 1999, (in Japanese).

[10] Teahan, W.J., PPMD+, Program, `http://www.cs.waikato.ac.nz/~wjt/software/ppm.tar.gz`.

[11] Willems, F.M.J., Shtarkov, Y.M., and Tjalkens, T.J., The context tree weighting method: basic properties. *IEEE Trans. Inform. Theory*, IT-41(3):653–664, May 1995.

Journal of Molecular Biology 1990 Oct.
Volume 215 (3)
pages 403-410

Basic Local Alignment Search Tool

Stephen F. Altschul[1], Warren Gish[1], Webb Miller[2]
Eugene W. Myers[3] and David J. Lipman[1]

[1]*National Center for Biotechnology Information*
National Library of Medicine, National Institutes of Health
Bethesda, MD 20894, U.S.A.

[2]*Department of Computer Science*
The Pennsylvania State University, University Park, PA 16802, U.S.A.

[3]*Department of Computer Science*
University of Arizona, Tucson, AZ 85721, U.S.A.

(Received 26 February 1990; accepted 15 May 1990)

A new approach to rapid sequence comparison, basic local alignment search tool (BLAST), directly approximates alignments that optimize a measure of local similarity, the maximal segment pair (MSP) score. Recent mathematical results on the stochastic properties of MSP scores allow an analysis of the performance of this method as well as the statistical significance of alignments it generates. The basic algorithm is simple and robust; it can be implemented in a number of ways and applied in a variety of contexts including straightforward DNA and protein sequence database searches, motif searches, gene identification searches, and in the analysis of multiple regions of similarity in long DNA sequences. In addition to its flexibility and tractability to mathematical analysis, BLAST is an order of magnitude faster than existing sequence comparison tools of comparable sensitivity.

1. Introduction

The discovery of sequence homology to a known protein or family of proteins often provides the first clues about the function of a newly sequenced gene. As the DNA and amino acid sequence databases continue to grow in size they become increasingly useful in the analysis of newly sequenced genes and proteins because of the greater chance of finding such homologies. There are a number of software tools for searching sequence databases but all use some measure of similarity between sequences to distinguish biologically significant relationships from chance similarities. Perhaps the best studied measures are those used in conjunction with variations of the dynamic programming algorithm (Needleman & Wunsch, 1970; Sellers, 1974; Sankoff & Kruskal, 1983; Waterman, 1984). These methods assign scores to insertions, deletions and replacements, and compute an alignment of two sequences that corresponds to the least costly set of such mutations. Such an alignment may be thought of as minimizing the evolutionary distance or maximizing the similarity between the two sequences compared. In either case, the cost of this alignment is a measure of similarity; the algorithm guarantees it is optimal, based on the given scores. Because of their computational requirements, dynamic programming algorithms are impractical for searching large databases without the use of a supercomputer (Gotoh & Tagashira, 1986) or other special purpose hardware (Coulson *et al.*, 1987).

Rapid heuristic algorithms that attempt to approximate the above methods have been developed (Waterman, 1984), allowing large databases to be searched on commonly available computers. In many heuristic methods the measure of similarity is not explicitly defined as a minimal cost set of mutations, but instead is implicit in the algorithm itself. For example, the FASTP program (Lipman & Pearson, 1985; Pearson & Lipman, 1988) first finds locally similar regions between two sequences based on identities but not gaps, and then rescores these regions using a measure of similarity between residues, such as a PAM matrix (Dayhoff *et al.*, 1978) which allows conservative replacements as well as identities to increment the similarity score. Despite their rather indirect approximation of minimal evolution measures, heuristic tools such as FASTP have been quite popular and have identified many distant but biologically significant relationships.

0022-2836/90/190403-08 $03.00/0

In this paper we describe a new method, BLAST† (Basic Local Alignment Search Tool), which employs a measure based on well-defined mutation scores. It directly approximates the results that would be obtained by a dynamic programming algorithm for optimizing this measure. The method will detect weak but biologically significant sequence similarities, and is more than an order of magnitude faster than existing heuristic algorithms.

2. Methods

(a) *The maximal segment pair measure*

Sequence similarity measures generally can be classified as either global or local. Global similarity algorithms optimize the overall alignment of two sequences, which may include large stretches of low similarity (Needleman & Wunsch, 1970). Local similarity algorithms seek only relatively conserved subsequences, and a single comparison may yield several distinct subsequence alignments; unconserved regions do not contribute to the measure of similarity (Smith & Waterman, 1981; Goad & Kanehisa, 1982; Sellers, 1984). Local similarity measures are generally preferred for database searches, where cDNAs may be compared with partially sequenced genes, and where distantly related proteins may share only isolated regions of similarity, e.g. in the vicinity of an active site.

Many similarity measures, including the one we employ, begin with a matrix of similarity scores for all possible pairs of residues. Identities and conservative replacements have positive scores, while unlikely replacements have negative scores. For amino acid sequence comparisons we generally use the PAM-120 matrix (a variation of that of Dayhoff *et al.*, 1978), while for DNA sequence comparisons we score identities +5, and mismatches −4; other scores are of course possible. A sequence segment is a contiguous stretch of residues of any length, and the similarity score for two aligned segments of the same length is the sum of the similarity values for each pair of aligned residues.

Given these rules, we define a maximal segment pair (MSP) to be the highest scoring pair of identical length segments chosen from 2 sequences. The boundaries of an MSP are chosen to maximize its score, so an MSP may be of any length. The MSP score, which BLAST heuristically attempts to calculate, provides a measure of local similarity for any pair of sequences. A molecular biologist, however, may be interested in all conserved regions shared by 2 proteins, not only in their highest scoring pair. We therefore define a segment pair to be locally maximal if its score cannot be improved either by extending or by shortening both segments (Sellers, 1984). BLAST can seek all locally maximal segment pairs with scores above some cutoff.

Like many other similarity measures, the MSP score for 2 sequences may be computed in time proportional to the product of their lengths using a simple dynamic programming algorithm. An important advantage of the MSP measure is that recent mathematical results allow the statistical significance of MSP scores to be estimated under an appropriate random sequence model (Karlin & Altschul, 1990; Karlin *et al.*, 1990). Furthermore, for any particular scoring matrix (e.g. PAM-120) one can estimate the frequencies of paired residues in maximal segments. This tractability to mathematical analysis is a crucial feature of the BLAST algorithm.

(b) *Rapid approximation of MSP scores*

In searching a database of thousands of sequences, generally only a handful, if any, will be homologous to the query sequence. The scientist is therefore interested in identifying only those sequence entries with MSP scores over some cutoff score S. These sequences include those sharing highly significant similarity with the query as well as some sequences with borderline scores. This latter set of sequences may include high scoring random matches as well as sequences distantly related to the query. The biological significance of the high scoring sequences may be inferred almost solely on the basis of the similarity score, while the biological context of the borderline sequences may be helpful in distinguishing biologically interesting relationships.

Recent results (Karlin & Altschul, 1990; Karlin *et al.*, 1990) allow us to estimate the highest MSP score S at which chance similarities are likely to appear. To accelerate database searches, BLAST minimizes the time spent on sequence regions whose similarity with the query has little chance of exceeding this score. Let a word pair be a segment pair of fixed length w. The main strategy of BLAST is to seek only segment pairs that contain a word pair with a score of at least T. Scanning through a sequence, one can determine quickly whether it contains a word of length w that can pair with the query sequence to produce a word pair with a score greater than or equal to the threshold T. Any such hit is extended to determine if it is contained within a segment pair whose score is greater than or equal to S. The lower the threshold T, the greater the chance that a segment pair with a score of at least S will contain a word pair with a score of at least T. A small value for T, however, increases the number of hits and therefore the execution time of the algorithm. Random simulation permits us to select a threshold T that balances these considerations.

(c) *Implementation*

In our implementations of this approach, details of the 3 algorithmic steps (namely compiling a list of high scoring words, scanning the database for hits, and extending hits) vary somewhat depending on whether the database contains proteins or DNA sequences. For proteins, the list consists of all words (w-mers) that score at least T when compared to some word in the query sequence. Thus, a query word may be represented by words in the list (e.g. for common w-mers using PAM scores) or by many. (One may, of course, insist that the w-mer in the query sequence be included in the word list irrespective of whether pairing the word with itself has a score of at least T.) For values of w and T that we found most useful (see below), there are typically on the order of 50 words in the list for every residue in the query sequence, e.g. 12,500 words for a sequence of length 250. If a little care is taken in programming, the list of words can be generated in time essentially proportional to the length of the list.

The scanning phase raised a classic algorithmic problem, i.e. search a long sequence for all occurrences of certain short sequences. We investigated 2 approaches. Simplified, the first works as follows. Suppose that $w = 3$ and map each word to an integer between 1 and 20^3.

† Abbreviations used: BLAST, blast local alignment search tool; MSP, maximal segment pair; bp, base-pair(s).

word can be used as an index into an array of size $20^4 = 160{,}000$. Let the ith entry of such an array point to the list of all occurrences in the query sequence of the ith word. Thus, as we scan the database, each database word leads us immediately to the corresponding hits. Typically, only a few thousand of the 20^4 possible words will be in this table, and it is easy to modify the approach to use far fewer than 20^4 pointers.

The second approach we explored for the scanning phase was the use of a deterministic finite automaton or finite state machine (Mealy, 1955; Hopcroft & Ullman, 1979). An important feature of our construction was to signal acceptance on transitions (Mealy paradigm) as opposed to on states (Moore paradigm). In the automaton's construction, this saved a factor in space and time roughly proportional to the size of the underlying alphabet. This method yielded a program that ran faster and we prefer this approach for general use. With typical query lengths and parameter settings, this version of BLAST scans a protein database at approximately 500,000 residues/s.

Extending a hit to find a locally maximal segment pair containing that hit is straightforward. To economize time, we terminate the process of extending in one direction when we reach a segment pair whose score falls a certain distance below the best score found for shorter extensions. This introduces a further departure from the ideal of finding guaranteed MSPs, but the added inaccuracy is negligible, as can be demonstrated by both experiment and analysis (e.g. for protein comparisons the default distance is 20, and the probability of missing a higher scoring extension is about 0·001).

For DNA, we use a simpler word list, i.e. the list of all contiguous w-mers in the query sequence, often with $w = 12$. Thus, a query sequence of length n yields a list of $n-w+1$ words, and again there are commonly a few thousand words in the list. It is advantageous to compress the database by packing 4 nucleotides into a single byte, using an auxiliary table to delimit the boundaries between adjacent sequences. Assuming $w \geq 11$, each hit must contain an 8-mer hit that lies on a byte boundary. This observation allows us to scan the database byte-wise and thereby increase speed 4-fold. For each 8-mer hit, we check for an enclosing w-mer hit; if found, we extend as before. Running on a SUN4, with a query of typical length (e.g. several thousand bases), BLAST scans at approximately 2×10^6 bases/s. At facilities which run many such searches a day, loading the compressed database into memory once in a shared memory scheme affords a substantial saving in subsequent search times.

It should be noted that DNA sequences are highly non-random, with locally biased base composition (e.g. A+T-rich regions), and repeated sequence elements (e.g. *Alu* sequences) and this has important consequences for the design of a DNA database search tool. If a given query sequence has, for example, an A+T-rich subsequence, or a commonly occurring repetitive element, then a database search will produce a copious output of matches with little interest. We have designed a somewhat *ad hoc* but effective means of dealing with these 2 problems. The program that produces the compressed version of the DNA database tabulates the frequencies of all 8-tuples. Those occurring much more frequently than expected by chance (controllable by parameter) are stored and used to filter "uninformative" words from the query word list. Also, preceding full database searches, a search of a sublibrary of repetitive elements is performed, and the locations in the query of significant matches are stored. Words generated by these regions are removed from the query word list for the full search. Matches to the sublibrary, however, are reported in the final output. These 2 filters allow alignments to regions with biased composition, or to regions containing repetitive elements to be reported, as long as adjacent regions not containing such features share significant similarity to the query sequence.

The BLAST strategy admits numerous variations. We implemented a version of BLAST that uses dynamic programming to extend hits so as to allow gaps in the resulting alignments. Needless to say, this greatly slows the extension process. While the sensitivity of amino acid searches was improved in some cases, the selectivity was reduced as well. Given the trade-off of speed and selectivity for sensitivity, it is questionable whether the gap version of BLAST constitutes an improvement. We also implemented the alternative of making a table of all occurrences of the w-mers in the database, then scanning the query sequence and processing hits. The disk space requirements are considerable, approximately 2 computer words for every residue in the database. More damaging was that for query sequences of typical length, the need for random access into the database (as opposed to sequential access) made the approach slower, on the computer systems we used, than scanning the entire database.

3. Results

To evaluate the utility of our method, we describe theoretical results about the statistical significance of MSP scores, study the accuracy of the algorithm for random sequences at approximating MSP scores, compare the performance of the approximation to the full calculation on a set of related protein sequences and, finally, demonstrate its performance comparing long DNA sequences.

(a) *Performance of BLAST with random sequences*

Theoretical results on the distribution of MSP scores from the comparison of random sequences have recently become available (Karlin & Altschul, 1990; Karlin *et al.*, 1990). In brief, given a set of probabilities for the occurrence of individual residues, and a set of scores for aligning pairs of residues, the theory provides two parameters λ and K for evaluating the statistical significance of MSP scores. When two random sequences of lengths m and n are compared, the probability of finding a segment pair with a score greater than or equal to S is:

$$1 - e^{-y}, \tag{1}$$

where $y = Kmn\ e^{-\lambda S}$. More generally, the probability of finding c or more distinct segment pairs, all with a score of at least S, is given by the formula:

$$1 - e^{-y} \sum_{i=0}^{c-1} \frac{y^i}{i!}. \tag{2}$$

Using this formula, two sequences that share several distinct regions of similarity can sometimes be detected as significantly related, even when no segment pair is statistically significant in isolation.

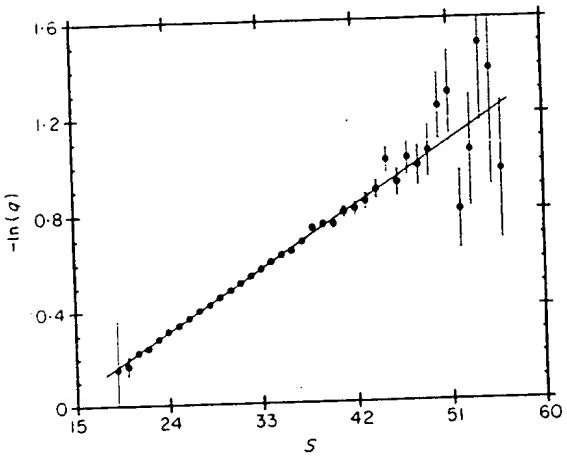


Figure 1. The probability q of BLAST missing a random maximal segment pair as a function of its score S.

While finding an MSP with a p-value of 0·001 may be surprising when two specific sequences are compared, searching a database of 10,000 sequences for similarity to a query sequence is likely to turn up ten such segment pairs simply by chance. Segment pair p-values must be discounted accordingly when the similar segments are discovered through blind database searches. Using formula (1), we can calculate the approximate score an MSP must have to be distinguishable from chance similarities found in a database.

We are interested in finding only segment pairs with a score above some cutoff S. The central idea of the BLAST algorithm is to confine attention to segment pairs that contain a word pair of length w with a score of at least T. It is therefore of interest to know what proportion of segment pairs with a given score contain such a word pair. This question makes sense only in the context of some distribution of high-scoring segment pairs. For MSPs arising from the comparison of random sequences, Dembo & Karlin (1991) provide such a limiting distribution. Theory does not yet exist to calculate the probability q that such a segment pair will fail to contain a word pair with a score of at least T. However, one argument suggests that q should depend exponentially upon the score of the MSP. Because the frequencies of paired letters in MSPs approaches a limiting distribution (Karlin & Altschul, 1990), the expected length of an MSP grows linearly with its score. Therefore, the longer an MSP, the more independent chances it effectively has for containing a word with a score of at least T, implying that q should decrease exponentially with increasing MSP score S.

To test this idea, we generated one million pairs of "random protein sequences" (using typical amino acid frequencies) of length 250, and found the MSP for each using PAM-120 scores. In Figure 1, we plot the logarithm of the fraction q of MSPs with score S that do not contain a word pair of length four with score at least 18. Since the values shown are subject to statistical variation, error bars represent one standard deviation. A regression line is plotted, allowing for heteroscedasticity (differing degrees of accuracy of the y-values). The correlation coefficient for $-\ln(q)$ and S is 0·999, suggesting that for practical purposes our model of the exponential dependence of q upon S is valid.

We repeated this analysis for a variety of word lengths and associated values of T. Table 1 shows the regression parameters a and b found for each instance; the correlation coefficient was always greater than 0·995. Table 1 also shows the implied percentage $q = e^{-(aS+b)}$ of MSPs with various scores that would be missed by the BLAST algorithm. These numbers are of course properly applicable only to chance MSPs. However, using a log-odds score matrix such as the PAM-120 that is based upon empirical studies of homologous proteins, high-scoring chance MSPs should resemble MSPs that reflect true homology (Karlin & Altschul, 1990). Therefore, Table 1 should provide a rough guide to the performance of BLAST on homologous as well as chance MSPs.

Based on the results of Karlin *et al.* (1990), Table 1 also shows the expected number of MSPs found when searching a random database of 16,000 length 250 protein sequences with a length 250 query. (These numbers were chosen to approximate the current size of the PIR database and the length of an average protein.) As seen from Table 1, only MSPs with a score over 55 are likely to be distinguishable from chance similarities. With $w = 4$ and $T = 17$, BLAST should miss only about a fifth of the MSPs with this score, and only about a tenth of MSPs with a score near 70. We will consider below the algorithm's performance on real data.

(b) *The choice of word length and threshold parameters*

On what basis do we choose the particular settings of the parameters w and T for executing BLAST on real data? We begin by considering the word length w.

The time required to execute BLAST is the sum of the times required (1) to compile a list of words that can score at least T when compared with a word from the query; (2) to scan the database for high scoring matches to words on this list); and (3) to extend hits to seek segment pairs with scores exceeding the cutoff. The time for the last of these tasks is proportional to the number of hits, which clearly depends on the parameters w and T. Given a random sequence model and a set of substitution scores, it is possible to calculate the probability that two random words of length w will have a score of at least T. The probability of a hit arising from an arbitrary pair of words in the query and the database, according to the random model and scores of the previous section, we have calculated these probabilities for a variety of parameter choices and recorded them in Table 2. For a given level of sensitivity (chance of missing a true MSP), one can ask what choice of w minimizes the

Table 1

The probability of a hit at various settings of the parameters w and T, and the proportion of random MSPs missed by BLAST

w	T	Probability of a hit $\times 10^5$	Linear regression $-\ln(q) = aS + b$		Implied % of MSPs missed by BLAST when S equals						
			a	b	45	50	55	60	65	70	75
3	11	253	0·1236	−1·005	1	1	0	0	0	0	0
	12	147	0·0875	−0·746	4	3	2	1	1	0	0
	13	83	0·0625	−0·570	11	8	6	4	3	2	2
	14	48	0·0463	−0·461	20	16	12	10	8	6	5
	15	26	0·0328	−0·353	33	28	23	20	17	14	12
	16	14	0·0232	−0·263	46	41	36	32	29	26	23
	17	7	0·0158	−0·191	59	55	51	47	43	40	37
	18	4	0·0109	−0·137	70	67	63	60	57	54	51
4	13	127	0·1192	−1·278	2	1	1	0	0	0	0
	14	78	0·0904	−1·012	5	3	2	1	1	0	0
	15	47	0·0686	−0·802	10	7	5	4	3	2	1
	16	28	0·0519	−0·634	18	14	11	8	6	5	4
	17	16	0·0390	−0·498	28	23	19	16	13	11	9
	18	9	0·0290	−0·387	40	35	30	26	22	19	17
	19	5	0·0215	−0·298	51	46	41	37	33	30	27
	20	3	0·0159	−0·234	62	57	53	49	45	41	38
5	15	64	0·1137	−1·525	3	2	1	1	0	0	0
	16	40	0·0882	−1·207	6	4	3	2	1	1	0
	17	25	0·0679	−0·939	12	9	6	4	3	2	2
	18	15	0·0529	−0·754	20	15	12	9	7	5	4
	19	9	0·0413	−0·608	29	23	19	15	13	10	8
	20	5	0·0327	−0·506	38	32	28	23	20	17	14
	21	3	0·0257	−0·420	48	42	37	32	29	25	22
	22	2	0·0200	−0·343	57	52	47	42	38	35	31
Expected no. of random MSPs with score at least S:					50	9	2	0·3	0·06	0·01	0·002

chance of a hit. Examining Table 1, it is apparent that the parameter pairs ($w = 3$, $T = 14$), ($w = 4$, $T = 16$) and ($w = 5$, $T = 18$) all have approximately equivalent sensitivity over the relevant range of cutoff scores. The probability of a hit yielded by these parameter pairs is seen to decrease for increasing w; the same also holds for different levels of sensitivity. This makes intuitive sense, for the longer the word pair examined the more information gained about potential MSPs. Maintaining a given level of sensitivity, we can therefore decrease the time spent on step (3), above, by increasing the parameter w. However, there are complementary problems created by large w. For proteins there are 20^w possible words of length w, and for a given level of sensitivity the number of words generated by a query grows exponentially with w. (For example, using the 3 parameter pairs above, a 30 residue sequence was found to generate word lists of size 96, 3561 and 40,939 respectively.) This increases the time spent on step (1), and the amount of memory required. In practice, we have found that for protein searches the best compromise between these considerations is with a word size of four; this is the parameter setting we use in all analyses that follow.

Although reducing the threshold T improves the approximation of MSP scores by BLAST, it also increases execution time because there will be more words generated by the query sequence and therefore more hits. What value of T provides a reasonable compromise between the considerations of sensitivity and time? To provide numerical data, we compared a random 250 residue sequence against the entire PIR database (Release 23·0, 14,372 entries and 3,977,903 residues) with T ranging from 20 to 13. In Figure 2 we plot the execution time (user time on a SUN4-280) *versus* the number of

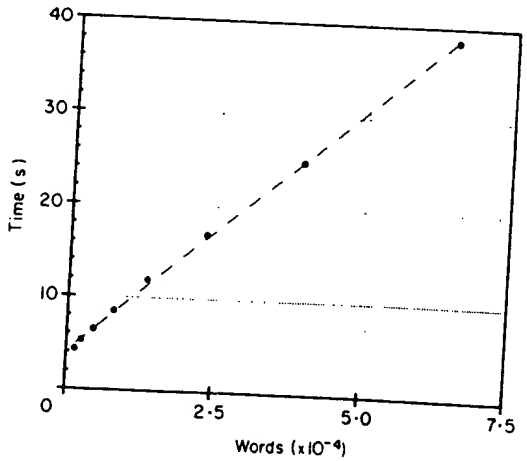


Figure 2. The central processing unit time required to execute BLAST on the PIR protein database (Release 23·0) as a function of the size of the word list generated. Points correspond to values of the threshold parameter T ranging from 13 to 20. Greater values of T imply fewer words in the list.

Table 2

The central processing unit time required to execute BLAST as a function of the approximate probability q *of missing an MSP with score* S

q (%)	CPU time (s)			
2	39	25	17	12
5	25	17	12	9
10	17	12	9	7
20	12	9	7	5
S:	44	55	70	90
p-value	1·0	0·8	0·01	10^{-5}

Times are for searching the PIR database (Release 23·0) with a random query sequence of length 250 using a SUN4-280. CPU, central processing unit.

words generated for each value of T. Although there is a linear relationship between the number of words generated and execution time, the number of words generated increases exponentially with decreasing T over this range (as seen by the spacing of x values). This plot and a simple analysis reveal that the expected-time computational complexity of BLAST is approximately $aW+bN+cNW/20^w$, where W is the number of words generated, N is the number of residues in the database and a, b and c are constants. The W term accounts for compiling the word list, the N term covers the database scan, and the NW term is for extending the hits. Although the number of words generated, W, increases exponentially with decreasing T, it increases only linearly with the length of the query, so that doubling the query length doubles the number of words. We have found in practice that $T = 17$ is a good choice for the threshold because, as discussed below, lowering the parameter further provides little improvement in the detection of actual homologies.

BLAST's direct tradeoff between accuracy and speed is best illustrated by Table 2. Given a specific probability q of missing a chance MSP with score S, one can calculate what threshold parameter T is required, and therefore the approximate execution time. Combining the data of Table 1 and Figure 2, Table 2 shows the central processing unit times required (for various values of q and S) to search the current PIR database with a random query sequence of length 250. To have about a 10% chance of missing an MSP with the statistically significant score of 70 requires about nine seconds of central processing unit time. To reduce the chance of missing such an MSP to 2% involves lowering T, thereby doubling the execution time. Table 2 illustrates, furthermore, that the higher scoring (and more statistically significant) an MSP, the less time is required to find it with a given degree of certainty.

(c) *Performance of BLAST with homologous sequences*

To study the performance of BLAST on real data, we compared a variety of proteins with other members of their respective superfamilies (Dayhoff, 1978), computing the true MSP scores as well as the BLAST approximation with word length four and various settings of the parameter T. Only with superfamilies containing many distantly related proteins could we obtain results usefully comparable with the random model of the previous section. Searching the globins with woolly monkey myoglobin (PIR code MYMQW), we found 178 sequences containing MSPs with scores between 50 and 80. Using word length four and T parameter 17, the random model suggests BLAST should miss about 24 of these MSPs; in fact, it misses 43. This poorer than expected performance is due to the uniform pattern of conservation in the globins, resulting in a relatively small number of high-scoring words between distantly related proteins. A contrary example was provided by comparing the mouse immunoglobulin κ chain precursor V region (PIR code KVMST1) with immunoglobulin sequences, using the same parameters as previously. Of the 33 MSPs with scores between 45 and 65, BLAST missed only two; the random model suggests it should have missed eight. In general, the distribution of mutations along sequences has been shown to be more clustered than predicted by a Poisson process (Uzzell & Corbin, 1971), and thus the BLAST approximation should, on average, perform better on real sequences than predicted by the random model.

BLAST's great utility is for finding high-scoring MSPs quickly. In the examples above, the algorithm found all but one of the 89 globin MSPs with a score over 80, and all of the 125 immunoglobulin MSPs with a score over 50. The overall performance of BLAST depends upon the distribution of MSP scores for those sequences related to the query. In many instances, the bulk of the MSPs that are distinguishable from chance have a high enough score to be found readily by BLAST, even using relatively high values of the T parameter. Table 3 shows the number of MSPs with a score above a given threshold found by BLAST when searching a variety of superfamilies using a variety of T parameters. In each instance, the threshold S is chosen to include scores in the borderline region, which in a full database search would include chance similarities as well as biologically significant relationships. Even with T equal to 18, virtually all the statistically significant MSPs are found in most instances.

Comparing BLAST (with parameters $w = 4$, $T = 17$) to the widely used FASTP program (Lipman & Pearson 1985; Pearson & Lipman, 1988) in its most sensitive mode ($ktup = 1$), we have found that BLAST is of comparable sensitivity, generally yields fewer false positives (high-scoring but unrelated matches to the query), and is over an order of magnitude faster.

(d) *Comparison of two long DNA sequences*

Sequence data exist for a 73,360 bp section of the human genome containing the β-like globin g

Table 3

The number of MSPs found by BLAST when searching various protein superfamilies in the PIR database (Release 22·0)

PIR code of query sequence	Superfamily searched	Cutoff score S	Number of MSPs with score at least S found by BLAST with T parameter set to							Number of MSPs in superfamily with score at least S
			22	20	19	18	17	16	15	
MYMQW	Globin	47	115	169	178	222	238	255	281	285
KVMST1	Immunoglobulin	47	153	155	155	156	156	157	158	158
OKBOG	Protein kinase	52	9	42	47	59	60	60	60	60
ITHU	Serpin	50	12	12	12	12	12	12	12	12
KYBOA	Serine protease	49	59	59	59	59	59	59	59	59
CCHU	Cytochrome *c*	46	81	91	91	96	98	98	98	98
FECF	Ferredoxin	44	22	23	23	24	24	24	24	24

MYMQW, woolly monkey myoglobin; KVMST1, mouse Ig κ chain precursor V region; OKBOG, bovine cGMP-dependent protein kinase; ITHU, human α-1-antitrypsin precursor; KYBOA, bovine chymotrypsinogen A; CCHU, human cytochrome *c*; FECF, *Chlorobium* sp. ferredoxin.

cluster and for a corresponding 44,595 bp section of the rabbit genome (Margot *et al.*, 1989). The pair exhibits three main classes of locally similar regions, namely genes, long interspersed repeats and certain anticipated weaker similarities, as described below. We used the BLAST algorithm to locate locally similar regions that can be aligned without introduction of gaps.

The human gene cluster contains six globin genes, denoted ε, $^G\gamma$, $^A\gamma$, η, δ and β, while the rabbit cluster has only four, namely ε, γ, δ and β. (Actually, rabbit δ is a pseudogene.) Each of the 24 gene pairs, one human gene and one rabbit gene, constitutes a similar pair. An alignment of such a pair requires insertion and deletions, since the three exons of one gene generally differ somewhat in their lengths from the corresponding exons of the paired gene, and there are even more extensive variations among the introns. Thus, a collection of the highest scoring alignments between similar regions can be expected to have at least 24 alignments between gene pairs.

Mammalian genomes contain large numbers of long interspersed repeat sequences, abbreviated *LINES*. In particular, the human β-like globin cluster contains two overlapped L1 sequences (a type of *LINE*) and the rabbit cluster has two tandem L1 sequences in the same orientation, both found 6000 bp in length. These human and rabbit L1 sequences are quite similar and their lengths make them highly visible in similarity computations. In all, eight L1 sequences have been cited in the human cluster and five in the rabbit cluster, but because of their reduced length and/or reversed orientation, the other published L1 sequences do not affect the results discussed below. Very recently, another piece of an L1 sequence has been discovered in the rabbit cluster (Huang *et al.*, 1990).

Evolution theory suggests that an ancestral gene cluster arranged as 5′-ε-γ-η-δ-β-3′ may have existed before the mammalian radiation. Consistent with this hypothesis, there are inter-gene similarities within the β clusters. For example, there is a region between human ε and $^G\gamma$ that is similar to a region between rabbit ε and γ.

We applied a variant of the BLAST program to these two sequences, with match score 5, mismatch score -4 and, initially, $w = 12$. The program found 98 alignments scoring over 200, with 1301 being the highest score. Of the 57 alignments scoring over 350, 45 paired genes (with each of the 24 possible gene pairs represented) and the remaining 12 involved L1 sequences. Below 350, inter-gene similarities (as described above) appear, along with additional alignments of genes and of L1 sequences. Two alignments with scores between 200 and 350 do not fit the anticipated pattern. One reveals the newly discovered section of L1 sequence. The other aligns a region immediately 5′ from the human β gene with a region just 5′ from rabbit δ. This last alignment may be the result of an intrachromosomal gene conversion between δ and β in the rabbit genome (Hardison & Margot, 1984).

With smaller values of w, more alignments are found. In particular, with $w = 8$, an additional 32 alignments are found with a score above 200. All of these fall in one of the three classes discussed above. Thus, use of a smaller w provides no essentially new information. The dependence of various values on w is given in Table 4. Time is measured in seconds on a SUN4 for a simple variant of BLAST that works with uncompressed DNA sequences.

Table 4

The time and sensitivity of BLAST on DNA sequences as a function of w

w	Time	Words	Hits	Matches
8	15·9	44,587	118,941	130
9	6·8	44,586	39,218	123
10	4·3	44,585	15,321	114
11	3·5	44,584	7345	106
12	3·2	44,583	4197	98

4. Conclusion

The concept underlying BLAST is simple and robust and therefore can be implemented in a number of ways and utilized in a variety of contexts. As mentioned above, one variation is to allow for gaps in the extension step. For the applications we have had in mind, the tradeoff in speed proved unacceptable, but this may not be true for other applications. We have implemented a shared memory version of BLAST that loads the compressed DNA file into memory once, allowing subsequent searches to skip this step. We are implementing a similar algorithm for comparing a DNA sequence to the protein database, allowing translation in all six reading frames. This permits the detection of distant protein homologies even in the face of common DNA sequencing errors (replacements and frame shifts). C. B. Lawrence (personal communication) has fashioned score matrices derived from consensus pattern matching methods (Smith & Smith, 1990), and different from the PAM-120 matrix used here, which can greatly decrease the time of database searches for sequence motifs.

The BLAST approach permits the construction of extremely fast programs for database searching that have the further advantage of amenability to mathematical analysis. Variations of the basic idea as well as alternative implementations, such as those described above, can adapt the method for different contexts. Given the increasing size of sequence databases, BLAST can be a valuable tool for the molecular biologist. A version of BLAST in the C programming language is available from the authors upon request (write to W. Gish); it runs under both 4·2 BSD and the AT&T System V UNIX operating systems.

W.M. is supported in part by NIH grant LM05110, and E.W.M. is supported in part by NIH grant LM04960.

References

Coulson, A. F. W., Collins, J. F. & Lyall, A. (1987). *Comput. J.* **30**, 420–424.

Dayhoff, M. O. (1978). Editor of *Atlas of Protein Sequence and Structure*, vol. 5, suppl. 3, Nat. Biomed. Res. Found., Washington, DC.

Dayhoff, M. O., Schwartz, R. M. & Orcutt, B. C. (1978). In *Atlas of Protein Sequence and Structure* (Dayhoff, M. O., ed.), vol. 5, suppl. 3, pp. 345–352. Nat. Biomed. Res. Found., Washington, DC.

Dembo, A. & Karlin, S. (1991). *Ann. Prob.* in the press.

Goad, W. B. & Kanehisa, M. I. (1982). *Nucl. Acids Res.* **10**, 247–263.

Gotoh, O. & Tagashira, Y. (1986). *Nucl. Acids Res.* **14**, 57–64.

Hardison, R. C. & Margot, J. B. (1984). *Mol. Biol. Evol.* **1**, 302–316.

Hopcroft, J. E. & Ullman, J. D. (1979). In *Introduction to Automata Theory, Languages, and Computation*, pp. 42–45, Addison-Wesley, Reading, MA.

Huang, X., Hardison, R. C. & Miller, W. (1990). *Comput. Appl. Biosci.* In the press.

Karlin, S. & Altschul, S. F. (1990). *Proc. Nat. Acad. Sci., U.S.A.* **87**, 2264–2268.

Karlin, S., Dembo, A. & Kawabata, T. (1990). *Ann. Stat.* **18**, 571–581.

Lipman, D. J. & Pearson, W. R. (1985). *Science*, **227**, 1435–1441.

Margot, J. B., Demers, G. W. & Hardison, R. C. (1989). *J. Mol. Biol.* **205**, 15–40.

Mealy, G. H. (1955). *Bell System Tech. J.* **34**, 1045–1079.

Needleman, S. B. & Wunsch, C. D. (1970). *J. Mol. Biol.* **48**, 443–453.

Pearson, W. R. & Lipman, D. J. (1988). *Proc. Nat. Acad. Sci., U.S.A.* **85**, 2444–2448.

Sankoff, D. & Kruskal, J. B. (1983) *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, Reading, MA.

Sellers, P. H. (1974). *SIAM J. Appl. Math.* **26**, 787–793.

Sellers, P. H. (1984). *Bull. Math. Biol.* **46**, 501–514.

Smith, R. F. & Smith, T. F. (1990). *Proc. Nat. Acad. Sci., U.S.A.* **87**, 118–122.

Smith, T. F. & Waterman, M. S. (1981). *Advan. Appl. Math.* **2**, 482–489.

Uzzell, T. & Corbin, K. W. (1971). *Science*, **172**, 1089–1096.

Waterman, M. S. (1984). *Bull. Math. Biol.* **46**, 473–500.

Edited by S. Brenner

BIOINFORMATICS

Vol. 15 no. 3 1999
Pages 194-202

Transformation distances: a family of dissimilarity measures based on movements of segments

Jean-Stephane Varré[1], Jean-Paul Delahaye[1] and Eric Rivals[2]

[1]Laboratoire d'Informatique Fondamentale de Lille (LIFL), UFR IEEA – Bât M3, 59655 Villeneuve d'Ascq Cedex, France and [2]Theoretische BioInformatik Deutsches Krebsforschungzentrum (DKFZ), Im Neuenheimer Feld 280, Heidelberg 69120, Germany

Received on October 6, 1998; revised on December 16, 1998; accepted on December 22, 1998

Abstract

Motivation: *Evolution acts in several ways on DNA: either by mutating a base, or by inserting, deleting or copying a segment of the sequence (Ruddle, 1997; Russell, 1994; Li and Grauer, 1991). Classical alignment methods deal with point mutations (Waterman, 1995), genome-level mutations are studied using genome rearrangement distances (Bafna and Pevzner, 1993, 1995; Kececioglu and Sankoff, 1994; Kececioglu and Ravi, 1995). The latter distances generally operate, not on the sequences, but on an ordered list of genes. To our knowledge, no measure of distance attempts to compare sequences using a general set of segment-based operations.*

Results: *Here we define a new family of distances, called* transformation distances, *which quantify the dissimilarity between two sequences in terms of segment-based events. We focus on the case where segment-copy, -reverse-copy and -insertion are allowed in our set of operations. Those events are weighted by their* description length, *but other sets of weights are possible when biological information is available. The transformation distance from sequence S to sequence T is then the* Minimum Description Length *among all possible scripts that build T knowing S with segment-based operations. The underlying idea is related to Kolmogorov complexity theory. We present an algorithm which, given two sequences S and T, computes exactly and efficiently the transformation distance from S to T. Unlike alignment methods, the method we propose does not necessarily respect the order of the residues within the compared sequences and is therefore able to account for duplications and translocations that cannot be properly described by sequence alignment. A biological application on Tnt1 tobacco retrotransposon is presented.*

Availability: *The algorithm and the graphical interface can be downloaded at http://www.lifl.fr/~varre/TD*

Contact: *{varre,delahaye}@lifl.fr, E.Rivals@dkfz-heidelberg.de*

Introduction

Evolution operates molecular alterations of two types: *point mutations*, namely insertion, deletion or substitution of single residues, and *segment-based modifications*: duplication, inversion, insertion, etc. of whole segments of the sequence. Genome level mutations operate also on large pieces of DNA and can thus be included in segment-based modifications. To our knowledge, no measure attempts to quantify dissimilarity by assessing segment-based differences, and by describing the differences between two sequences with an edit-script containing such segment-based operations.

Sequence comparison is usually performed on similar parts of the sequences, like structurally or functionally related domains of proteins. Even if they correspond to complete biological entities like a whole gene or a protein, entire sequences are not compared, or only in case of high similarity. With such restrictions, one misses some biological meaningful information written in the molecules.

Certain alignment oriented methods take into account the existence of 'segment' of similarity. Morgenstern *et al.* (1996) proposed a segment-based alignment method which aligns pairs of direct segments having local similarities and excludes regions of low similarity from the alignment. Schöniger and Waterman (1992) found a dynamic programming algorithm to include in a classical alignment parts where the sequences are aligned in reverse order. These solutions are alignment oriented: the segments put into correspondence always respect the overall order of the positions in the sequences.

In our approach, this order is disregarded. We do not want to restrict our attention to 'alignable' segment similarities, but also consider a wider class of segment operations like: duplication, inversion, or translocation. We use a different definition of similarity and this leads to a different class of problems.

Other approaches, called genome rearrangement distances, quantify segment-based evolution through the minimal number of operations needed to transform a 'chromosome', i.e. an ordered list of genes, into another 'chromosome', the same list of genes in another order. Several operations were considered (often one operation per distance): reversals (Hannenhalli and Pevzner, 1995), transpositions (Bafna and Pevzner, 1998), translocations (Hannenhalli, 1996), block interchanges (Christie, 1996) and the syntenic distance which is applied to (unordered) set of genes (Ferretti *et al.*, 1996). In most contexts, those methods do not apply directly to sequences, but to given lists of labels each one representing a gene. The costs are user parameters. Most of those distances are computationally expensive. In our framework, segments which are rearranged have to be discovered. Moreover, we propose to weight more precisely the operations.

We propose a new measure which evaluates segment-based dissimilarity between two sequences: the source S and the target T. This measure relates to the process of constructing the target sequence T with *segment operations*. The construction starts with the empty string and proceeds from left to right by adding segments, one segment per operation. A list of operations is called a *script*. Three *types of segment operations* are considered: the *copy* adds segments that are contained in the source sequence S, the *reverse copy* adds segments that are contained in S in reverse order, and the *insertion* adds segments that are not necessarily contained in S. For the sake of clarity, we call the insertion of a segment in general, a *type of operation*, while the insertion of the segment, say 'agtct', is an *operation* because it is completely specified. The measure depends on a parameter that is the *Minimum Factor Length*; it is the minimum length of the segments that can be copied or reverse copied. A positive weight is assigned to each operation and the weight of a script is the sum of the weights of its operations. Depending on the number of common segments between S and T, there exist several scripts that construct the target T. The *minimal scripts* are all scripts of minimum weight and the *transformation distance* (TD) is the weight of a minimal script. This defines a precise optimization problem which we solve in this work. We would like to emphasize that with another set of types of operations, one can use the same generic definition of transformation distance. Thus, the transformation distance generalizes in a family of measures of distance between sequences.

How are operations weighted? In our framework, unit cost is meaningless (see next section). From the biological point of view, a satisfactory probabilistic model does not exist that applies to segment operations on a sequence and would allow us to derive weights. Therefore we apply another idea and follow the principle of parsimony. The principle of parsimony is justified by the more general Minimum Description Length Principle (MDLP) (Li and Vitányi, 1997; Rissanen, 1989). We adopt the MDLP and weight each operation by its *description length*. A generic description scheme is associated with a type of operation. For the copy, the description requires a 2-bits code to distinguish the type of operation, an offset between the locations of the segment in S and in T, and the length of the segment. To a reverse copy corresponds the same description but with another 2-bits code for a reverse copy. For an insertion, one needs the 2-bits code for an insertion, the length of the segment and the sequence of segment. Some examples are given in the next section.

Description length is not an arbitrary way of weighting. In fact, it seems natural in the absence of specific knowledge: it represents the quantity of information necessary to describe an operation. This property has its underlying in the Algorithmic Information Theory (Kolmogorov, 1965). This theory suggests that the description length is the fairest weighting scheme that one could use a priori. The use of information theory is advocated by Yockey (1992). For an introduction to the AIT, we refer the reader to the book of Li and Vitányi (1997); an explanation of its use to 'weight events' in computer sequence analysis can be read in Rivals *et al.* (1996).

Additionally to this definition, we present a polynomial time algorithm to compute exactly the transformation distance according to the definition given above. For this, we consider a weighted graph of all possible scripts. We demonstrate that minimal scripts correspond to the shortest paths from a source node to a sink node, representing respectively the left- and right-end of the target sequence. Taking into account the relative weights of different segment-operations, we use properties of some non-optimal scripts which allow us to exclude them from the graph. The subsequent decrease in the size of the graph results in a practicable algorithm. Moreover, we provide an efficient implementation together with a user-friendly interface, and make them available to the community through our web-site.

In a study of the family of sequences of Tnt1 Tobacco retrotransposons, the TD reveals the presence of segments duplications and segments re-orderings in some parts of the sequences. These sequences are clearly not alignable and therefore only restricted comparisons are possible with alignment methods.

The transformation distances

In the introduction, we defined precisely the TD we use and wrote that it can be generalized to adopt other sets of types of operations. We detail this point here, and explain why the description length is a reasonable choice for the weights, although not the only possible one.

The set of types of segment operations

First, the set must contain an insertion. When constructing T, one may need to add a segment which does not occur in the source S. Clearly, the set must enclose a type of operation which enables this: it is the insertion. In addition to the insertion, most of the other operations can be included: those used in genome rearrangement distances, but also the du- or multiplication of a segment which could be in tandem or not, and the deletion.

Some remarks must be stated. First, the set we choose allows detecting most of the events mentioned above. For example, our 'copy' can displace a segment from the source in the target: special cases of this are transpositions and block interchanges. The difference is that in our framework, transpositions are not given a different weight than block interchanges. Nevertheless, transpositions and block interchanges can be detected. Second, the more types of operations that are considered, the higher the number of possible scripts: this will increase the complexity of the problem. Third, deletion is a very peculiar operation which requires that an already inserted segment could be modified in a second operation: this corresponds to a construction which is not left to right. This also requires a more complex algorithm.

Operations weights

We mentioned that unit costs are unsuitable in our framework. Indeed, a script containing a single insertion of the segment T itself always exists. With unit costs, this script would be minimal. Thus, when choosing a weighting function, one must let the insertion of a 'long' segment of S cost more than a copy of it, so that the TD is able to reveal sequence relationship. The description length fulfills this property because the segment of a copy is available in the source sequence S, while it must be encoded explicitly in an insertion, and also because the description length increases with the segment length.

The study of the construction of a target sequence is the object of the AIT in the case of general sequences. This theory suggests that for the measure we define here, description length is the fairest a priori weights one can choose. This is because the process of constructing a sequence using a set of operations has intrinsic properties, whatever the type of sequences. Because of these properties, the minimum description length measures the information content of an operation, and here operations are potential modifications of the genetic sequences. Detailed explanations are beyond the scope of this paper. In his book Yockey (1992) reviews the characteristics of information theory and concludes that using it should benefit sequence comparison.

The weighting function can be adapted according to biological knowledge. For instance, the weights can be multiplied by a given coefficient if one wants to favor copies instead of reverse copies. This does not change the definition, nor the algorithm.

Remarks and justification

Interpretation of the TD and of a script. The TD and the associated minimal scripts are clearly not an attempt to view evolution as a computer program. First in biological evolution, the source and the target sequences derived from a common ancestor sequence, say A. Our model is a simplification in that it considers a process linking directly the source to the target and which does not exist. Nevertheless, a minimal script suggests (1) what parts of the sequences are common to S and T, (2) which of those parts may have been rearranged in their relative order, and (3) possible operations for those rearrangements which could have happened between A and S or A and T.

Minimum Factor Length parameter. There is a limit of the length of the segment, below which inserting a segment costs more than copying it if it occurs in S. One may interpret this limit, saying that below, it is unclear whether the segment appeared by convergence or divergence. Therefore, all segments shorter than the *Minimum Factor Length* are systematically added using insertions.

A script as a program. In fact, 'executing' the script builds the sequence T. The script is a program which outputs T when S is supplied as data. The definition of the script is a restricted form of the definition of a *program* given by Kolmogorov in the AIT (Li and Vitányi, 1997). The conditional Kolmogorov complexity of a string x relatively to a string y, denoted $K(x|y)$, is the length of a shortest binary program which, on a universal Turing machine, outputs x if y is furnished as an auxiliary input data. $K(x|y)$ measures the minimal amount of information required to generate x knowing y by any effective process. This defines the algorithmic information distance. The transformation distance approximates of the relative Kolmogorov complexity of T knowing S: as our 'machine' only allows three instructions (copy, reverse-copy and insertion) it is not universal. Therefore, the transformation distance does not consider all programs but only those limited to these 3 instructions. On the other hand, unlike the general algorithmic distance, the transformation distance is computable.

Properties. The computation of the transformation distance does not depend on the way we read sequences (5′ to 3′ or 3′ to 5′). This stems from the way we encode target positions. The transformation distance is not symmetrical ($d(S,T) \neq d(T,S)$). It is intrinsic to our definition: the way of describing S from T is not necessarily the same that the one of describing T from S. When a symmetrical distance is required, one can use the following definition: ($d(S,T)$ +

$d(T,S))/2$. The transformation distance does not satisfy the triangular inequality. However, we have made some experiments and it seems that, in practice, the triangular inequality is often satisfied.

Encoding scripts. To be comparable, descriptions have to be written in the same language. We use the binary language because efficient encoding procedures are known. As DNA is made up from $4(=2^2)$ possible bases, each of them might be encoded over 2 (the exponent) bits. A n-bases long sequence is thus encoded over $2n$ bits. The number of bits required for representing an integer l is $\lceil \log_2(|l|+1) \rceil$. When the item can be either positive or negative, we add one bit for the sign. One needs a 2-bits code to encode each type of operation ($2 = \log_2(3)$).

Figure 1 gives an example of a script and its weights. The first line (insertion(UGUGCA)) has a weight of 2+12+3: $2 = \lceil \log_2(3) \rceil$ is the number of bits required to encode the type of the operation, $12 = 2 \times 6$ is the number of bits to encode the segment explicitly, $3 = \lceil \log_2(6+1) \rceil$ is the number of bits to encode the length of the segment. The fourth line (copy(27, 113, 8)) has a weight of 2+1+7+4: as for the insertion 2 is for encoding the type of the operation, $4 = \lceil \log_2(8+1) \rceil$ is for the length, $7 = \lceil \log_2((113-27)+1) \rceil$ is the number of bits required to encode the offset between the locations of the segment in S (113) and in T (27), 1 is for the sign of the offset because it can be negative (e.g. the copy of h).

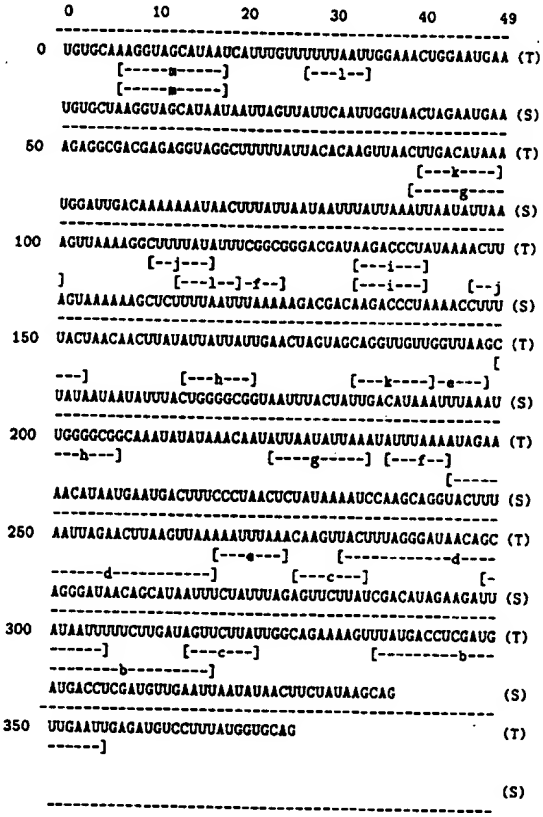
A script is defined by its copies. We remark that a script is entirely defined by specifying which copies or reverse-copies it contains. This means that insertions can be deduced from those information alone. Indeed, the insertions must provide the segments of T which are not brought by the copies, i.e. the complementary parts. In the algorithm, searching for a script is equivalent to a search for a combination of segment pairs that can be copied.

Algorithm

This section describes the algorithm we have designed to compute the transformation distance. We show that the minimal script corresponds to the shortest path from a source node to a sink node in a weighted directed graph we define below.

Factors

Let us call a *factor* a pair of segments, one segment from each sequence (source and target), such that the two segments are either identical or the first segment is the reverse complement of the second one. The set of all factors is denoted F. A factor is specified by the triplet (p,q,l): its starting positions in the target sequence (p), in the source sequence (q) and its length (l). We define a relation $<_0$ on the set of all factors such that



	Operation	Cost
	insertion(UGUGCA)	2+12+3
m	copy(6, 6, 13)	2+1+1+4
	insertion(CAUUUGUU)	2+16+4
l	copy(27, 113, 8)	2+1+7+4
	insertion(GGAAA...ACAAGUUAAC)	2+110+6
k	copy(90, 183, 10)	2+1+7+4
	insertion(AGUUAAAAGG)	2+20+4
j	copy(110, 146, 8)	2+1+6+4
	insertion(UU UCGGCGGGAC GAU)	2+30+4
i	copy(133, 133, 9)	2+1+1+4
	insertion(UAAAACUU...UUGGUUAAG)	2+114+6
h	copy(199, 164, 9)	2+1+6+4
	insertion(CA AAUAUAUAAA CAAU)	2+32+5
g	copy(224, 89, 12)	2+1+8+4
	insertion(U)	2+2+1
f	copy(237, 118, 8)	2+1+7+4
	insertion(UAGAA AAUUAGAACU UAAGUUAA)	2+46+5
e	copy(268, 190, 9)	2+1+7+4
	insertion(CAA GU)	2+10+3
d	copy(282, 244, 25)	2+1+6+5
	insertion(UUC UUGAU)	2+16+4
c	copy(315, 277, 9)	2+1+6+4
	insertion(UGGCAG AAAAGU)	2+24+4
b	copy(336, 298, 21)	2+1+6+5
	insertion(GAG AUGUCCUUUA UGGUGCAG)	2+42+5
		709

Fig. 1. An example of the computation of the transformation distance onto two RNA sequences. The target and the source are displayed aligned on their first residue. Common segments are denoted by letters within brackets. The associated script is shown in the table below: one line per operation and with the corresponding weight (i.e. number of necessary bits for this operation).

for two factors $f=(p,q,l)$ and $g=(p',q',l')$, $f <_0 g$ holds if $p+l \leq p'$, i.e. if f strictly precedes g in the target sequence.

To search for all factors, we use the algorithm of Leung *et al.* which is able to discover exact but also point mutated repeats. To find factors, we apply this algorithm onto the text formed by the concatenation of T, S and the reverse-complement of S. We implemented the algorithm to allow substitutions inside factors without modifying the weight function, i.e. without adding a penalty. For simplicity, we describe it in the case of exact factors.

Script graph

As written above, a script is entirely specified by its copy operations (from now we do not distinguish copy and reverse copy and simply write 'copy'), i.e. when the set of its copied factors is known. Those factors in a script do not overlap in the target (see definition, condition (i)), this is a simple consequence of the construction process. Additionally, if many factors are concurrent for some segment of the target, i.e. if they are overlapping, at least one of them must be copied. In other words, an optimal script cannot 'forget' a factor that can be copied (condition (ii)). We introduce the definition of a *factor* script which is a script that fulfills those two properties.

Definition A *factor script FS* is a set of factors such that:
(i) for all $f,g \in FS, f <_0 g$ or $g <_0 f$
(ii) and for all f,g in FS such that $f <_0 g$ implies $\nexists\, h \in$ F–FS such that $f <_0 h <_0 g$.

We now define the script graph which is the basic structure for the computation of the transformation distance.

Definition *A and Z are respectively source and sink nodes of the graph. f and g are two factors of F, the set of all factors.* The *script* graph $G=(V,E)$ is defined by:

- $V = \mathrm{F} \cup \{A,Z\}$,
- $(f,g) \in E$ *iff*:
  (1) $f <_0 g$,
  (2) and $\nexists\, h \in \mathrm{F}$ such that $f <_0 h <_0 g$,
- $(A,f) \in E$ *iff* $\nexists\, h \in \mathrm{F}$ such that $h <_0 f$,
- $(f,Z) \in E$ *iff* $\nexists\, h \in \mathrm{F}$ such that $f <_0 h$

The script graph is a directed graph where factors are the nodes. Two nodes joined by an edge represent successive copies in a script and their factors fulfill the $<_0$ relation. Each edge is oriented from the leftmost factor towards the rightmost one (we build the target sequence from left to right). Additionally, we define a source node A and a sink node Z which serve respectively as the beginning and end of any factor script. A can be viewed as the factor (0,0,0) and Z as the factor $(|T|,|S|,0)$. A path from A to Z represents a selection of factors and as such defines a unique factor script.

In order to compute the description length of each script (path) we assign costs to edges and vertices. The cost of a vertex is the cost of the copy of the factor it represents. The cost of an edge is the cost of the insertion needed between the copied segments, i.e. the source and target nodes of this edge. The cost of a path is obtained by adding costs of all edges and vertices on this path.

Proposition *The computation of the shortest path going from source node A to the sink node Z gives the minimum description length script.*

Sketch of the proof: each path of the graph is clearly a script because it combines copies and insertion operations such that the points 1 and 2 of the definition are verified. Each possible script is represented by a path in the graph. In fact, all possible copies operations are included in the graph and the set of successors of a node f contains exactly all the nodes which can be reached from f. The cost of a path is the description length of the associated script: indeed, the cost of a path is the sum of the costs of the insertions operations (edges) and the copies operations (nodes) it contains.

Time and space complexity

Computing the set of all factors with the Leung *et al.* (1991) algorithm is known to be efficient. Statistical studies have shown that its complexity increases almost linearly with the sequence lengths. Computing the shortest script is achieved in $O(Card(V)+Card(E))$ with the algorithm *Dag-Shortest-Path* presented in chapter 25.4 of Cormen *et al.* (1990). Let us denote n the length of longest sequence among the target sequence and the source sequence. There are at most n^3 segments between S and T, and therefore as much vertices in the script graph. As a complete graph over n^3 vertices has less than n^6 edges, the computation of the transformation distance requires less than $O(n^6)$ units of time. This complexity is for worst cases and is a loose approximation. Nevertheless, in practice, the computation of the complete script graph is too inefficient to be applied on long sequences (more than 100 kb). It is the space requirement that prevents the computation.

Practicable algorithm

In practice, the size of the complete script graph grows dramatically for long sequences (of more than 100 kb), and particularly in the case of similar sequences because they share numerous factors. We studied the properties of copies operations that cannot belong to the minimal script. The corresponding factors can thus be removed from the graph. So without defining it here, we implement the *compact script graph* which encloses only 'interesting' copies and reverse copies. The above-mentioned properties and the definition of this compact graph cannot be detailed here for the sake of shortness. Moreover for implementation matter, only maximal factors, those that are not sub-factors of another factor, are included in the graph at run time. It is then possible to create their sub-factors only when necessary. Acceleration of

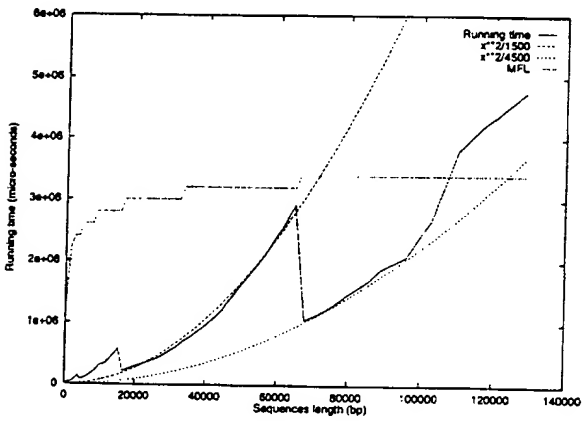


Fig. 2. Running times of the distance computation versus sequence length (on a PC Pentium 233 computer).

the computation time is illustrated hereunder. Table 1 reports the number of factors and the construction time for the script graph and the compact script graph. Only the compact script graph allows computing in little time the transformation distance on long sequences (more than 500 kb).

Figure 2 illustrates the variation of the running time in function of the sequence length. The source and target sequences are the tobacco's and rice's chloroplast genomes. To show the influence of the sequence length, we compute the TD for longer and longer prefixes of these sequences. For instance, the point of the curve with abscissa $x = 80\ 000$ corresponds to the running time for the prefixes of length 80 000 bps of the source and the target. The Minimal Factor Length parameter was set to $\lceil \log_2 (|T|+1) \rceil$ where $|T|$ is the length of T and it varies with the sequences lengths (it is also plotted on the graphic). The vertical drops of the plain line curve denote an acceleration because the number of nodes decreased. They correspond to losses of factors when the minimal factor length reaches a new discrete value. In fact, they relate to drops in the MFL curve. With the present implementation, the time requirement is short even for long sequences: around 5 s for 130 kb.

Implementation

The algorithm is implemented in C and available at http://www.lifl.fr/~varre/TD. As shown on Figure 3, a user-friendly graphical interface (implemented with Tcl/Tk) allows to compute all against all comparisons for a set of sequences and to visualize each comparison.

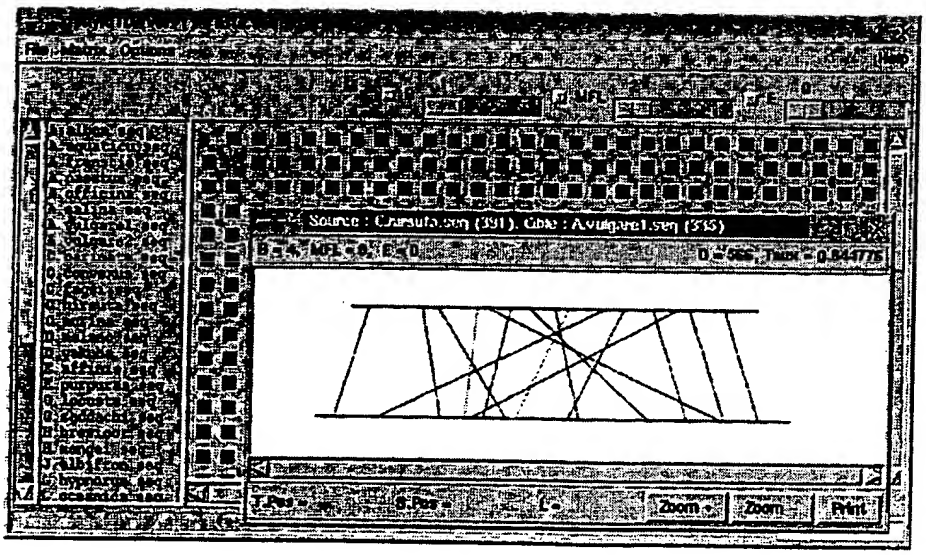


Fig. 3. Example of computation of the transformation distance for a set of sequences. One can display the actual distance and the corresponding minimal script between two sequences by clicking on the corresponding square in the matrix.

Table 1. Construction time and number of factors included in the graph for both the script graph (SG) and the compact script graph (CSG). For the *Rice, Tobacco, O. sinensis* and *E. gracilis*, the sequences used are the complete chloroplast genomes. Sequences *HIV1* and *HIV2* are two clones of the HIV type 1 genome (accession numbers U34603 and U34604). *Bac. Subtilis 1* and *9* are parts of the bacillus subtilis complete genome (accession numbers Z99104 and Z99112). *C.E. 1* and *C.E. 2* are two clones of Caenorhabditis elegans (accession numbers Z98856 and Z92860). Running times have been computed on a PC Pentium 166MMX. A '–' indicates that the program exhausts the computer memory

		Number of factors		Running time (s)	
	Sequence length (kb)	SG	CSG	SG	CSG
Rice and tobacco	140	10 963	818	75	6.9
E. gracilis and *O. sinensis*	140	1683	209	18.4	16
HIV1 and HIV2	10	8578	111	–	7.6
Bac. Subtilis 1 and *Bac. Subtilis 9*	250	18 144	6	–	18
C.E. 1 and *C.E. 2*	630	20 172	5147	–	151

Results and discussion

We applied the TD to investigate the evolution of the TnT1 tobacco retrotransposon. The results illustrate the usefulness of the TD and its larger applicability compared to alignment methods: the TD allows finding segments duplications and re-orderings which may result from evolutionary events.

Families of Tnt1 tobacco retrotransposon

The problem considered here is the evolution of Tnt1 tobacco's retrotransposon, and specifically the evolution of its Long Terminal Repeat (LTR). The material is a set of 140 sequences of this LTR taken out of seven species of tobacco. The LTR feature is the following from 5 to 3: the RT box, the linker, then the U3 and R boxes. It has been suggested that the high mobility of Tnt1 may require rapid evolution (Casacuberta *et al.*, 1995).

We computed all pairwise comparisons for the 140 sequences (the running time was less than 1 h). Figure 4 shows the representative comparison of the retrotransposons of *Tobacco sylvestris* (top horizontal line) and *Tobacco tomentosiformis* (bottom horizontal line). The 5′ and 3′ ends feature each 3 parallel vertical bars that link the segments shared by both sequences. It shows that those regions corresponding respectively to the RT + linker and to the R box are well conserved, and thus also well aligned (this is observed on all comparisons). On the opposite in the middle part of the sequence, in the U3 region, one sees 6 vertical bars which intersect each other, suggesting that the order of the corresponding segments has changed during divergence of those sequences. Moreover, two vertical bars have the same end-point and then refer to the same segment in *T. tomentosiformis* sequence, while they point to different segments (i.e. end-points) in *T. sylvestris*: this segment may have been duplicated during evolution. Such segment relations observed in many pairwise comparisons simply prevent correct alignment. Those results are consistent with the analysis of Casacuberta *et al.* in which they suggested that the RT + linker and R boxes were conserved, while they suspected rearrangements inside the U3 box. However providing evidence for the latter was nearly impossible as it required studying by eye all pairwise alignments. The TD algorithm allowed us to detect and visualize some rearrangements events automatically, suggesting that the TD is a useful and complementary approach to classical alignment strategies.

Conclusion

This work provides a new measure, the transformation distance, for comparing genetic sequences and an efficient algorithm to compute it. The application to Tobacco retrotransposons TnT1 points out types of sequence relationships which are undetectable with alignments. Indeed, it detects segments duplication and re-ordering which usually prevent correct alignments. This argues for the usefulness of the transformation distances as an alternative tool for the investigation of sequences relationships. Compared to alignments, our work shows that the concepts of the Algorithmic Information Theory may be useful to suggest practical approaches and effective algorithms in a biological context. Among others, wider applications of the transformation distance are: phylogenies, sequences clustering and analysis, and investigation of segment-based evolution.

The use of other weights and/or other sets of operations than those studied here yields variants of the TDs that may include biological knowledge. Thus, the general idea of our method is susceptible to various specifications to be explored.

We suggest that the transformation distance may be particularly appropriate to investigate the evolution of RNA sequences, where the palindromic segments may correspond to elements of the secondary structure. The TD favors conservation of such segments and would thus better account for the secondary structure. The study of the phylogeny of isopods based on mitochondrial RNA sequences is in progress.

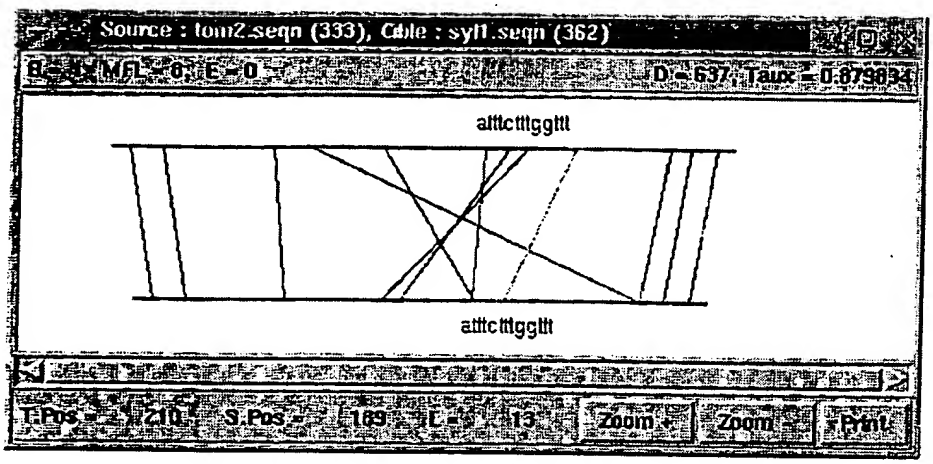


Fig. 4. Comparison of Tnt1 tobacco retrotransposon of tobacco tomentosiformis (as source) and tobacco sylvestris (as target).

Acknowledgments

We would like to thank Samantha Vernhettes, Helene Chiapello and Marie-Angèle Grandbastien (INRA Versailles, http://www.inra.fr/Versailles/BIOCEL) for Tnt1 retrotransposon data and very interesting discussions. The authors are also grateful to Didier Bouchon and Alice Michel (Génétique et Biologie des Populations de Crustacés, UMR CNRS 6556, http://wwwumr6556.univ-poitiers.fr) for their data and numerous useful comments, and to Dominique Anxolabéhére (Modélisation de la dynamique des populations d'éléments transposables, Dynamique du Génome et Evolution, Institut Jacques Monod, Paris VII, http://www.ijm.jussieu.fr) for interesting discussions. E. R. thanks E. Bornberg and M.-L. Muiras (DKFZ) for their careful reading of the manuscript.

The authors also wish to thank the referees for their sharp and helpful comments.

References

Bafna,V. and Pevzner,P. (1993) Genome rearrangements and sorting by reversals. In *Proceedings of the 34th FOCS*, IEEE, pp. 148–157.

Bafna,V. and Pevzner,P. (1995) Sorting permutations by transpositions. In *6th Symposium on Discrete Algorithms SODA*, pp. 614–621.

Bafna,V. and Pevzner,P. (1998) Sorting by transpositions. *SIJDM: SIAM J. Discrete Math.*, **11.**

Casacuberta,J., Vernhettes,S. and Grandbastien,M.A. (1995) Sequence variability within the tobacco retrotransposon Tnt1 population. *EMBO J.*, **14,** 2670–2678.

Christie,D.A. (1996) Sorting permutations by block-interchanges. *Inf. Process. Lett.*, **60,** 165–169.

Cormen,T.H., Leiserson,C.E. and Rivest,R.L. (1990) *Introduction to Algorithms*. MIT Press.

Doolittle,R.F. (1981) Similar amino acid sequences: chance or common ancestry? *Science*, **214,** 149–159.

Ferretti,V., Nadeau,J.H. and Sankoff,D. (1996) Original synteny, combinatorial pattern matching. In: Hirschberg,D.S. and Myers,E.W. (eds), *7th Annual Symposium*, Lecture Notes in Computer Science, **1075,** 159–167. Springer, Berlin.

Grumbach,S. and Tahi,F. (1995) Compression et compréhension de séquences nucléotidiques. *Technique et Science Informatique*, **14,** 217–233.

Hannenhalli,S. and Pevzner,P. (1995) Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*. Las Vegas, Nevada, pp. 178–189.

Hannenhalli,S. (1996) Polynomial-time algorithm for computing translocation distance between genomes. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, Vol. 71.

Hillis,D.M., Moritz,C. and Mable,B.K. (1996) *Molecular Systematics.* Sinauer Associates Inc.

Kececioglu,J. and Sankoff,D. (1994) Efficient bounds for oriented chromosome inversion. In *5th Symposium on Combinatorial Pattern Matching*, pp. 307–325.

Kececioglu,J. and Ravi,R. (1995) Of mice and men: algorithms of evolutionary distances between genomes with translocations. In *6th Symposium on Discrete Algorithms SODA*, pp. 604–613.

Kolmogorov,A.N. (1965) Three approaches to the quantitative definition of information. *Probl. Inf. Transmiss.*, **1,** 1–7.

Leung,M.Y., Blaisdell,B.E., Burge,C. and Karlin,S. (1991) An efficient algorithm for identifying matches with errors in multiple long molecular sequences. *J. Molec. Biol.*, **221,** 1367–1378.

Li,W.H. and Graur,D. (1991) *Fundamentals of Molecular Evolution.* Sinauer Associates Inc.

Li,M. and Vitányi,P.M.B. (1997) *An Introduction to Kolmogorov Complexity and Its Applications.* 2nd Edn. Springer, New York.

Morgenstern,B., Dress,A. and Werner,T. (1996) Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl Acad. Sci. USA*, **93**, 12098–12103.

Rissanen,J. (1989) *Stochastical Complexity and Statistical Inquiry*. World Scientific.

Rivals,E., Dauchet,M., Delahaye,J.P. and Delgrange,O. (1996) Compression and genetic sequences analysis. *Biochimie*, **78**.

Ruddle,F.H. (1997) Vertebrate genome evolution – the decade ahead. *Genomics*, **46**, 171–173.

Russel,R.B. (1994) Domain insertion. *Protein Eng.*, **7**, 1407–1410.

Sankoff,D. and Blanchette,M. (1998) Multiple genome rearrangements and breakpoint phylogeny. *J. Computat. Biol.*, **5**, 555–570.

Schöniger,M. and Waterman,M.S. (1992) A local algorithm for DNA sequence alignment with inversions. *Bull. Math. Biol.*, **54**, 521–536.

Varré,J.S., Delahaye,J.P. and Rivals,E. (1997) *The Transformation Distance*. Genome Informatics Workshop, Tokyo, Japan.

Waterman,M.S. (1995) *Introduction to Computational Biology*. Chapman and Hall.

Yockey,H.P. (1992) *Information Theory and Molecular Biology*. Cambridge University Press, Cambridge.

Proc. Natl. Acad. Sci. USA
Vol. 90, pp. 5873–5877, June 1993
Evolution

Applications and statistics for multiple high-scoring segments in molecular sequences

(sequence comparison/pattern recognition/molecular features/statistical significance)

SAMUEL KARLIN[†] AND STEPHEN F. ALTSCHUL[‡§]

[†]Department of Mathematics, Stanford University, Stanford, CA 94305; and [‡]National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD 20894

Contributed by Samuel Karlin, March 24, 1993

ABSTRACT **Score-based measures of molecular-sequence features provide versatile aids for the study of proteins and DNA. They are used by many sequence data base search programs, as well as for identifying distinctive properties of single sequences. For any such measure, it is important to know what can be expected to occur purely by chance. The statistical distribution of high-scoring segments has been described elsewhere. However, molecular sequences will frequently yield several high-scoring segments for which some combined assessment is in order. This paper describes the statistical distribution for the sum of the scores of multiple high-scoring segments and illustrates its application to the identification of possible transmembrane segments and the evaluation of sequence similarity.**

The study of molecular-sequence data can be assisted by statistical methods of sequence analysis. Among the aims of such study is the discovery of patterns relevant to genomic organization, nucleic acid processing, protein folding, and biochemical function as well as their evolutionary developments. A region of unusual amino acid composition in a protein sequence may correlate with a specific biological function. Similarly, the conservation over evolutionary time of segments shared by different proteins may provide clues to structure and function.

Among the tools for detecting interesting regions in protein sequences are score-based methods. These assign appropriate positive numerical values to amino acids likely to be found within the type of region sought and negative values to residues unlikely to occur. Since scores permit differentiation between residues, they engender more sensitive analyses than do measures that consider only simple matching. Scores have been used to locate transmembrane or significantly hydrophobic segments, DNA-binding domains, and regions of concentrated charge (1). They are also employed to identify similar regions shared by two or more protein or DNA molecules and are at the core of many sequence data base search programs (2–4).

A crucial question for any given score-based or other measure applied to molecular sequences is what can be expected to occur purely by chance. Empirical statistical studies can be based upon sequence data collections (1, 5–8) or upon permutations of sample sequences (9, 10). In addition, analytic statistical results can afford calculable criteria for the evaluation of sequences and can elucidate the function of the parameters in the measures to which they apply (11). They provide means for recognizing outliers, for developing contrasting sequence classifications, and for comparing different data sets in a consistent manner.

The greatest limitation on the analytic approach is the difficulty of deriving statistical distributions for any but the simplest sequence measures. Among those that have yielded to analysis are ones based on runs of a given residue type, allowing for a specified number or proportion of mismatches (12–14). More recently, a theory has been developed for characterizing unusual sequence patterns, defined with reference to general scoring systems (15–18). Scores may be based on residue biochemical or physical properties or, in the case of sequence comparison, on residue similarities. Specifically, the theory describes the asymptotic extremal distribution of high aggregate segment scores as well as the letter composition of high-scoring segments.

In this paper we consider several natural extensions of score-based measures. An important such extension is the sum of the r greatest segment scores. This measure is appropriate when there may be several distinct segments of a given type within a protein or DNA sequence (e.g., transmembrane segments). Also, for sequence comparisons, the existence of insertions or deletions can break an alignment into several pieces, and the sum of their scores can be an appropriate measure of local sequence similarity. From this consideration there arises the problem of "consistency" for high-scoring segment pairs: the requirement that multiple pairs be combinable into a single "gapped" alignment. We discuss below how this constraint affects the distribution of the sum statistic. The use of these statistics will be illustrated with several examples.

The Statistical Theory for High-Scoring Segments

Given a molecular sequence, we assume that scores are assigned to the various sequence elements and study the statistical behavior of the segment (of whatever length) with greatest aggregate score. In this section we review briefly the theory for such maximal-segment scores (15–18). The basic themes of this theory are visible in the various extensions that follow.

In the simple "independence" random sequence model we employ, the elements of a sequence are chosen independently from an alphabet of a letters with respective probabilities $p_1, \ldots, p_a$. A DNA sequence, for example, would have $a = 4$, and a protein sequence using the standard alphabet would have $a = 20$. Theory exists for the more complicated case of Markov-dependent sequences but will not be discussed here (17). A score s_i is assigned to each type of letter. For proteins these scores may be based, for example, on physicochemical or structure-related properties such as charge, size, hydrophobicity, and helix-forming potential. The maximal segment of a sequence is defined as that contiguous string of letters with greatest aggregate score. The random distribution for the score S of this segment can be expressed by using three parameters, which are described below.

The publication costs of this article were defrayed in part by page charge payment. This article must therefore be hereby marked "*advertisement*" in accordance with 18 U.S.C. §1734 solely to indicate this fact.

Abbreviations: p.d.f., probability density function; PIR, Protein Identification Resource.

[§]To whom reprint requests should be addressed.

A necessary assumption for the following theory is that the expected score per letter $\Sigma_i p_i s_i$ be negative. [Scores based on likelihood ratios (15) always satisfy this condition.] Were the expected score positive, the maximal segment would always tend to be virtually the entire sequence, so such a scoring system would not be of much use for identifying unusual regions. The existence of at least one positive score along with the previous condition implies that there is always a unique positive solution $x = \lambda$ to the equation $\Sigma_i p_i e^{s_i x} = 1$. The parameter λ may be thought of as a natural scale for the scoring system employed.

A second parameter K, for which an explicit but more complex formula is available, is also readily calculated from the scores s_i and their background probabilities p_i (15–18). The final relevant parameter is the length N of the random sequence from which the maximal segment is drawn. The statistical theory is then most simply expressed in terms of the normalized score $S' = \lambda S - \ln KN$. For large N, the tail probability (Prob) that S' is greater than or equal to x is well approximated by the formula

$$\text{Prob}(S' \geq x) \approx 1 - \exp(-e^{-x}). \qquad [1]$$

For sequence comparison, the theory has a parallel development. Scores s_{ij} are now assigned not to individual letters but to pairs of letters. Given two sequences, the maximal-segment pair is simply that pair of equal-length segments, one from each sequence, which when aligned have maximal aggregate score S. The expected score per residue pair must still be negative, and the formulas for λ and K are the same as before. The main difference is that the "search space size" parameter N becomes the product of the lengths of the two sequences being compared. A number of conditions must hold for $\text{Prob}(S' \geq x)$ to converge to formula 1 for large N (A. Dembo, S.K., and O. Zeitouni, unpublished data), but this formula is always conservative—i.e., it provides an upper bound on the desired probability.

For single-sequence protein analysis, scores appropriate for the detection of transmembrane segments and DNA-binding domains have been described (1, 6, 19). For sequence comparison, a wide range of scoring systems have been proposed (11, 20–28), and segment-pair scores underpin the BLAST data base search programs (2, 29, 30). These are examples where the basic theory finds direct application. In many cases, however, more sophisticated scoring methods, such as those studied in this paper, are appropriate.

The Statistical Theory for Multiple High-Scoring Segments

Sometimes a single molecule will contain multiple regions with a common property of biological interest, or two molecules will share several quite similar regions. For example, a protein may have several distinct transmembrane segments or regions of concentrated charge. Two proteins may share a number of regions of conserved secondary structure, separated by loops of variable length and composition. When this is the case, seeking the single highest-scoring region can discard much valuable information. We therefore consider the scores $S_1, \ldots, S_r$ of the r highest-scoring distinct segments. Statistics for these random variables are most conveniently written using the normalized scores $S'_k = \lambda S_k - \ln KN$. For large N, the joint probability density function (p.d.f.) for $S'_1, \ldots, S'_r$ is well approximated by the formula

$$f(x_1, \ldots, x_r) = \exp\left(-e^{-x_r} - \sum_{k=1}^{r} x_k\right), \qquad [2]$$

where $x_1 \geq x_2 \geq \cdots \geq x_r$. The distribution of any function of the S'_k may be calculated from this distribution. The simplest application is to calculate the tail probability that S'_r is greater than or equal to x; integration yields

$$\text{Prob}(S'_r \geq x) \approx 1 - \exp(-e^{-x}) \sum_{k=0}^{r-1} \frac{e^{-kx}}{k!}, \qquad [3]$$

which is a generalization of formula 1.

Of greater interest and utility is the distribution of the sum of the r highest normalized scores $T_r = S'_1 + \cdots + S'_r$. From formula 2 and some algebraic manipulation, one may show that for large N, the p.d.f. for T_r approaches

$$f(t) = \frac{e^{-t}}{r!(r-2)!} \int_0^\infty y^{r-2} \exp(-e^{(y-t)/r})\, dy. \qquad [4]$$

All moments of this distribution may be calculated by means of Laplace transforms. The mean is given by $r(1 + \gamma - \Sigma_{k=1}^{r} 1/k)$, where $\gamma \approx 0.577$ is Euler's constant, and is well approximated by $r(1 - \ln r) - \frac{1}{2}$. The variance is $r^2(\pi^2/6 - \Sigma_{k=1}^{r} 1/k^2) + r$, which is approximately $2r - \frac{1}{2}$.

To obtain the tail probability that $T_r \geq x$, one must integrate Eq. 4 for t from x to infinity. This double integral is easily calculated numerically, and a program for the purpose in the C programming language is available from the authors. In the limit of large x, this tail probability behaves as

$$\text{Prob}(T_r \geq x) \approx \frac{e^{-x} x^{r-1}}{r!(r-1)!}. \qquad [5]$$

Applications of these results will be given below.

Consistently Ordered Segment Pairs in Sequence Alignments

Two proteins may share distinct, homologous domains that need not retain the same relative order. More often, however, separate high-scoring segment pairs arise from insertions or deletions within a matching region. In the context of pairwise sequence comparison, one may wish to exclude the former possibility and consider only the latter; this simultaneously excludes from analysis cases that do not fit the biological model and increases the statistical significance of those segment pair sets that do.

Requiring that a collection of high-scoring segment pairs be consistent with a single alignment including gaps imposes a certain "geometry" on the pairs that so far has not been taken into account. For a given high-scoring segment pair i, let (x_i, y_i) indicate the midpoints of its constituent segments within their respective sequences. A necessary condition for combining several segment pairs into a single consistent alignment is that for any two pairs i and j, $x_i < x_j$ if and only if $y_i < y_j$. We will call a set of segment pairs "consistently ordered" if it satisfies this condition.

The random variable T_r from the previous section can be written as $\lambda(\Sigma_{k=1}^{r} S_k) - \ln K^r \to \ln N^r$. The last term may be understood as correcting for the N^r different possible sets of starting positions for the r segment pairs whose scores are the S_k. (Remember that for pairwise sequence comparison, N is the product of the lengths of the sequences being compared.) If we require a set of r segment pairs to be consistently ordered before allowing their scores to be combined, we effectively divide the size of the possible solution space by $r!$. Therefore, if T^*_r is the greatest value attainable as the sum of normalized scores S'_k from r distinct and consistently ordered segment pairs, $T^*_r + \ln r!$ has a p.d.f. approaching that of Eq. 4 for large N.

This analysis can be extended to more restrictive constraints on the relationship of combined segment pairs. Between pairs, for example, one may allow gaps within each

Table 1. High-scoring segments of the *D. virilis* sevenless protein (34) [Protein Identification Resource (PIR) code A35774] with their associated scores and *P* values

Segment	Positions	Score (normalized score)	*P* value	
			Segment	Sum
LVLAIIAPAAIVSSCVLALVLV	2141–2162	67 (4.4)	0.012	0.012
FLVTGHGGISTILIANLLLLLLLSL	116–140	55 (2.5)	0.080	0.0035
ISAPIIVALLAL	466–477	38 (−0.2)	0.71	0.0053

Segment scores are based on the transmembrane scores from ref. 1.

sequence of only some maximum size. The appropriate statistics then depend upon the extent to which the space of possible solutions is reduced.

The measure presented in this section combines gaps and scores in a natural manner. One drawback of this measure, however, is that so long as a gap is permitted at all, no premium is placed on a short as opposed to a long one. A statistical problem that remains open is the random distribution of scores from optimal alignments that include gaps and for which length-dependent gap costs are assessed (4, 31). While numerical studies have been conducted on the statistics of such scoring systems (7, 8), they have resisted complete analysis to date.

Further Results Involving Consistent Ordering

One question similar in spirit but different in detail from those considered above is how many distinct segments can be expected with score at least x. This question is most easily answered by using the composite parameter $y = KN e^{-\lambda x}$. For large N and for x sufficiently large that y is not much greater than 1, the number of distinct segments whose score is at least x is then approximately Poisson distributed with parameter y (15–18). In other words, the probability of observing exactly k such segments is approximately $e^{-y}y^k/k!$. The probability of observing at least r segments with score at least x is calculated by summing this quantity for k from r to infinity.

In the case of sequence alignments, we now wish to impose the additional requirement of consistent ordering. The most natural question concerns the probability that there are at least r distinct and consistently ordered segment pairs all with score at least x. The desired probability arises if each term of the infinite sum is multiplied by the probability that a set of k segment pairs contains a consistently ordered subset of size at least r. For large N this probability can be seen to approach $R_{k,r}/k!$, where $R_{k,r}$ is the number of permutations of the integers 1 to k that contain an increasing subsequence of length at least r. Thus, the formula for the desired probability becomes

$$e^{-y} \sum_{k=r}^{\infty} \frac{y^k R_{k,r}}{k!^2}. \qquad [6]$$

To employ formula 6 effectively, one must be able to calculate $R_{k,r}$ for at least the first several k values greater than or equal to r. When $r \leq 4$, general formulas are available for $R_{k,r}$ (32). Moreover, for all r, various combinatorial facts about permutations (33) suffice to prove that $R_{r,r} = 1$; $R_{r+1,r} = r^2 + 1$, and $R_{r+2,r} = (r^4 + 2r^3 + r^2 + 2r + 6)/2$. Specific but increasingly complicated formulas may be derived for successive terms. However, the first three terms just given should be sufficient for most purposes.

Examples

To illustrate the use of sum statistics, we first consider the sevenless protein from the fruit fly *Drosophila virilis* (34). This molecule is a tyrosine kinase receptor required for embryogenesis of the eye; it is known to have one and is suspected to have two transmembrane domains (35). We analyzed the molecule for transmembrane segments, using scores derived for this purpose by Karlin and Brendel (1). The three highest-scoring segments of the protein are shown in Table 1, arranged in decreasing order of score. For this analysis, the relevant statistical parameters may be calculated as $\lambda = 0.159$, $K = 0.21$, and $N = 2594$. The single highest-scoring segment, consisting of residues 2141–2162, has a normalized score of 4.4, which by formula 1 corresponds to a *P* value of 0.012. The second highest normalized score (for residues 116–140) is 2.5, corresponding to a *P* value of 0.08. Neither of these segments in isolation may be considered significant at the 99% level. However, as shown in Table 1, when analyzed in unison, the *P* value for the sum of their scores drops to 0.0035. Successive high-scoring segments (i.e., those other than the top-scoring two) do not improve the overall result. The two segments identified as statistically significant by this method are the putative transmembrane domains described in the original paper (34).

As a second example, we analyze the human serotonin receptor (36) for transmembrane segments. This molecule is a member of the large family of guanine nucleotide-binding protein-coupled receptors, which generally contain seven transmembrane segments, accounting for roughly half of the complete protein. The large proportion of hydrophobic residues within these proteins render concentrations of such

Table 2. High-scoring segments of the human serotonin receptor (36) (PIR code S07343), with their associated scores and *P* values

Segment	Positions	Score (normalized score)	*P* value	
			Segment	Sum
VITSLLLGTLIFCAVLGNACVVAAIAL	(37) 37–63 (62)	66 (3.5)	0.031	0.031
LGIIMGTFILCWLPFFIVALVL	(345) 346–367 (366)	65 (3.4)	0.034	0.0036
ALISLTWLIGFLISI	(152) 154–168 (177)	46 (1.2)	0.26	0.0019
IYSTFGAFYIPLLLMLVL	(191) 196–213 (216)	41 (0.6)	0.42	0.0011
LIGSLAVTDLMVSVLVLPMAAL	(74) 74–95 (98)	38 (0.3)	0.53	0.00064
LFIALDVLCCTSSILHLCAIAL	(110) 111–132 (134)	31 (−0.5)	0.81	0.00056
LLGAII	(378) 379–384 (402)	26 (−1.1)	0.95	0.00061

Segment scores are based on the transmembrane scores from ref. 1. Next to the position numbers representing the extent of each high-scoring segment are given in parentheses those for the corresponding putative transmembrane segment as specified in SWISS-PROT (38).

Table 3. High-scoring segment pairs, with their associated scores and *P* values, from a comparison of the chicken gene X protein (39) (PIR code DXCH) and the fowlpox virus antithrombin III homolog (40) (PIR code WMVZF3)

Segment pair	Positions	Score (normalized score)	*P* value	
			Segment pair	Sum
VYLPQMKIEEXYNLTSVLMALGMTDLF	125–151			
YLP E L L G DLF		52 (7.6)	4.7×10^{-4}	4.7×10^{-4}
LYLPKFELEDDVDLKDALIHMGCNDLF	44–70			
SANLTGISSAESLKISQAVHGAFMELSEDGIEMAGST	154–190			
S L GIS L I E G E A T		49 (6.7)	1.2×10^{-3}	4.2×10^{-6}
SGELVGISDTKTLRIGNIRQKSVIKVDEYGTEAASVT	72–108			
RADHPFLFLIKHNPTNTIVYFGRY	206–229			
A PF FL T G		46 (5.8)	3.1×10^{-3}	5.9×10^{-8}
KANVPFMFLVADVQTKIPLFLGIF	123–146			

Segment pair scores are calculated using the PAM-120 scoring matrix (11, 20). Amino acid identities are echoed on the central line of each alignment.

amino acids more difficult to distinguish from chance (cf. ref. 37).

Using the same transmembrane scores as before (1), the seven best segments of the human serotonin receptor are shown in Table 2, arranged in decreasing order of score. Here the relevant parameters are $\lambda = 0.114$, $K = 0.14$, and $N = 421$. The single highest-scoring segment consists of residues 37–63 and has a normalized score of 3.5, corresponding by formula 1 to a *P* value of 0.031. Therefore, neither this nor any of the other high-scoring segments may be considered, in isolation, particularly surprising. When several of the highest-scoring segments are analyzed in unison, however, the situation changes. As shown in Table 2, *P* values for the sum of the *r* highest segment scores continue to drop until $r = 6$, at which point the cumulative normalized score of 8.4 has a probability less than 6×10^{-4} of having occurred by chance. Further segments do not improve the overall result. It should be noted that the statistics for sums of high segment scores described above are valid only in the limit of large *N*. For a protein as short as the one in this example, they are inaccurate for $r > 2$. Nevertheless, even the sum of just the two highest segment scores provides good evidence that one is dealing with a multisegment transmembrane protein. Applications of the sum statistic to long DNA sequences will be discussed elsewhere.

Finally, to illustrate the use and potential power of sum statistics applied to pairwise sequence comparison, we consider an analysis of the chicken gene *X* protein (39) and the fowlpox virus antithrombin III homolog (40). When compared by using the PAM-120 amino acid substitution matrix (11, 20), three high-scoring segment pairs emerge (Table 3). Given this scoring system and the amino acid frequencies of the two sequences, $\lambda = 0.314$, $K = 0.17$, and $N = 34336$, yielding corrected scores of 7.6, 6.7, and 5.8 for the three alignments. From formula 1, the associated *P* values for these alignments are all less than 0.004, which is generally considered significant. However, such similarities are frequently uncovered in protein data base searches, in which tens of thousands of pairwise comparisons are typically performed (2). In such a multitrial context, *P* values near 10^{-6} generally are necessary before statistical significance may be claimed. None of the individual alignments shown in Table 3 achieve such significance, and any single one of them could easily arise by chance in a search of current protein sequence data bases (38, 41). This is no longer the case, however, when the three segment pairs are considered together. The sum of their normalized scores is 20.1, which for $r = 3$ corresponds to a *P* value of 5.9×10^{-8}, easily significant even in the context of a large data base search. Notice as well that the three segment pairs shown in Table 3 are consistently ordered. (In fact, the three pairs are in almost perfect alignment.) When this is taken as an *a priori* requirement for invoking a sum, the *P* value drops even further, to 1.2×10^{-8}. Thus, the ability to calculate statistics for the combined scores of distinct segment pairs can greatly increase the sensitivity of sequence comparison tools.

S.F.A. thanks Dr. John Spouge for helpful conversations and Dr. Warren Gish for programming assistance. We appreciate valuable comments on the manuscript from Dr. Volker Brendel. S.K. was supported in part by National Institutes of Health Grants GM39907-02 and GM10452-26 and National Science Foundation Grant DMS86-06244.

1. Karlin, S. & Brendel, V. (1992) *Science* **257**, 39–49.
2. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. (1990) *J. Mol. Biol.* **215**, 403–410.
3. Pearson, W. R. & Lipman, D. J. (1988) *Proc. Natl. Acad. Sci. USA* **85**, 2444–2448.
4. Smith, T. F. & Waterman, M. S. (1981) *J. Mol. Biol.* **147**, 195–197.
5. Collins, J. F., Coulson, A. F. W. & Lyall, A. (1988) *Comput. Appl. Biosci.* **4**, 67–71.
6. Karlin, S., Bucher, P., Brendel, V. & Altschul, S. F. (1991) *Annu. Rev. Biophys. Biophys. Chem.* **20**, 175–203.
7. Mott, R. (1992) *Bull. Math. Biol.* **54**, 59–75.
8. Smith, T. F., Waterman, M. S. & Burks, C. (1985) *Nucleic Acids Res.* **13**, 645–656.
9. Altschul, S. F. & Erickson, B. W. (1985) *Mol. Biol. Evol.* **2**, 526–538.
10. Fitch, W. M. (1983) *J. Mol. Biol.* **163**, 171–176.
11. Altschul, S. F. (1991) *J. Mol. Biol.* **219**, 555–565.
12. Arratia, R., Gordon, L. & Waterman, M. S. (1986) *Ann. Stat.* **14**, 971–993.
13. Arratia, R. & Waterman, M. S. (1989) *Ann. Probab.* **17**, 1152–1169.
14. Karlin, S. & Ost, F. (1988) *Ann. Probab.* **16**, 535–563.
15. Karlin, S. & Altschul, S. F. (1990) *Proc. Natl. Acad. Sci. USA* **87**, 2264–2268.
16. Dembo, A. & Karlin, S. (1991) *Ann. Probab.* **19**, 1737–1755.
17. Karlin, S. & Dembo, A. (1992) *Adv. Appl. Probab.* **24**, 113–140.
18. Karlin, S., Dembo, A. & Kawabata, T. (1990) *Ann. Stat.* **18**, 571–581.
19. Brendel, V., Bucher, P., Nourbakhsh, I. R., Blaisdell, B. E. & Karlin, S. (1992) *Proc. Natl. Acad. Sci. USA* **89**, 2002–2006.
20. Dayhoff, M. O., Schwartz, R. M. & Orcutt, B. C. (1978) in *Atlas of Protein Sequence and Structure*, ed. Dayhoff, M. O. (Natl. Biomed. Res. Found., Washington, DC), Vol. 5, Suppl. 3, pp. 345–352.
21. Feng, D. F., Johnson, M. S. & Doolittle, R. F. (1985) *J. Mol. Evol.* **21**, 112–125.
22. Gonnet, G. H., Cohen, M. A. & Benner, S. A. (1992) *Science* **256**, 1443–1445.
23. Henikoff, S. & Henikoff, J. G. (1992) *Proc. Natl. Acad. Sci. USA* **89**, 10915–10919.
24. Jones, D. T., Taylor, W. R. & Thornton, J. M. (1992) *Comput. Appl. Biosci.* **8**, 275–282.

25. McLachlan, A. D. (1971) *J. Mol. Biol.* **61**, 409–424.
26. Schwartz, R. M. & Dayhoff, M. O. (1978) in *Atlas of Protein Sequence and Structure*, ed. Dayhoff, M. O. (Natl. Biomed. Res. Found., Washington, DC), Vol. 5, Suppl. 3, pp. 353–358.
27. States, D. J., Gish, W. & Altschul, S. F. (1991) *Methods* **3**, 66–70.
28. Wilbur, W. J. (1985) *Mol. Biol. Evol.* **2**, 434–447.
29. Altschul, S. F. & Lipman, D. J. (1990) *Proc. Natl. Acad. Sci. USA* **87**, 5509–5513.
30. Gish, W. & States, D. J. (1993) *Nature Genet.* **3**, 266–272.
31. Sellers, P. H. (1984) *Bull. Math. Biol.* **46**, 501–514.
32. Gessel, I. M. (1990) *J. Combinat. Theory A* **53**, 257–285.
33. Knuth, D. E. (1973) *The Art of Computer Programming* (Addison–Wesley, Reading, MA), Vol. 3, pp. 48–72.
34. Michael, W. M., Bowtell, D. D. & Rubin, G. M. (1990) *Proc. Natl. Acad. Sci. USA* **87**, 5351–5353.
35. Simon, M. A., Bowtell, D. D. & Rubin, G. M. (1989) *Proc. Natl. Acad. Sci. USA* **86**, 8333–8337.
36. Kobilka, B. K., Frielle, T., Collins, S., Yang-Feng, T., Kobilka, T. S., Francke, U., Lefkowitz, R. J. & Caron, M. G. (1987) *Nature (London)* **329**, 75–79.
37. Karlin, S., Brendel, V. & Bucher, P. (1992) *Mol. Biol. Evol.* **9**, 152–167.
38. Bairoch, A. & Boeckmann, B. (1992) *Nucleic Acids Res.* **20**, 2019–2022.
39. Heilig, R., Perrin, F., Gannon, F., Mandel, J. L. & Chambon, P. (1980) *Cell* **20**, 625–637.
40. Tomley, F., Binns, M., Campbell, J. & Boursnell, M. (1988) *J. Gen. Virol.* **69**, 1025–1040.
41. Barker, W. C., George, D. G., Mewes, H. W. & Tsugita, A. (1992) *Nucleic Acids Res.* **20**, 2023–2026.

BLAST HELP MANUAL

DESCRIPTION

This document describes the WWW BLAST interface.

BLAST (Basic Local Alignment Search Tool) is the heuristic search algorithm employed by the programs blastp, blastn, blastx, tblastn, and tblastx; these programs ascribe significance to their findings using the statistical methods of Karlin and Altschul (1990, 1993) with a few enhancements. The BLAST programs were tailored for sequence similarity searching -- for example to identify homologs to a query sequence. The programs are not generally useful for motif-style searching. For a discussion of basic issues in similarity searching of sequence databases, see Altschul et al. (1994).

The five BLAST programs described here perform the following tasks:

blastp compares an amino acid query sequence against a protein sequence database;

blastn compares a nucleotide query sequence against a nucleotide sequence database;

blastx compares the six-frame conceptual translation products of a nucleotide query sequence (both strands) against a protein sequence database;

tblastn compares a protein query sequence against a nucleotide sequence database dynamically translated in all six reading frames (both strands).

tblastx compares the six-frame translations of a nucleotide query sequence against the six-frame translations of a nucleotide sequence database.

BLAST Search parameters

HISTOGRAM
Display a histogram of scores for each search; default is yes. (See parameter H in the BLAST Manual).

DESCRIPTIONS
Restricts the number of short descriptions of matching sequences reported to the number specified; default limit is 100 descriptions. (See parameter V in the manual page). See also EXPECT and CUTOFF.

ALIGNMENTS
Restricts database sequences to the number specified for which high-scoring segment pairs (HSPs) are reported; the default limit is 50. If more database sequences

than this happen to satisfy the statistical significance threshold for reporting (see EXPECT and CUTOFF below), only the matches ascribed the greatest statistical significance are reported. (See parameter B in the BLAST Manual).

EXPECT

The statistical significance threshold for reporting matches against database sequences; the default value is 10, such that 10 matches are expected to be found merely by chance, according to the stochastic model of Karlin and Altschul (1990). If the statistical significance ascribed to a match is greater than the EXPECT threshold, the match will not be reported. Lower EXPECT thresholds are more stringent, leading to fewer chance matches being reported. Fractional values are acceptable. (See parameter E in the BLAST Manual).

CUTOFF

Cutoff score for reporting high-scoring segment pairs. The default value is calculated from the EXPECT value (see above). HSPs are reported for a database sequence only if the statistical significance ascribed to them is at least as high as would be ascribed to a lone HSP having a score equal to the CUTOFF value. Higher CUTOFF values are more stringent, leading to fewer chance matches being reported. (See parameter S in the BLAST Manual). Typically, significance thresholds can be more intuitively managed using EXPECT.

MATRIX

Specify an alternate scoring matrix for BLASTP, BLASTX, TBLASTN and TBLASTX. The default matrix is BLOSUM62 (Henikoff & Henikoff, 1992). The valid alternative choices include: PAM40, PAM120, PAM250 and IDENTITY. No alternate scoring matrices are available for BLASTN; specifying the MATRIX directive in BLASTN requests returns an error response.

STRAND

Restrict a TBLASTN search to just the top or bottom strand of the database sequences; or restrict a BLASTN, BLASTX or TBLASTX search to just reading frames on the top or bottom strand of the query sequence.

FILTER

Mask off segments of the query sequence that have low compositional complexity, as determined by the SEG program of Wootton & Federhen (Computers and Chemistry, 1993), or segments consisting of short-periodicity internal repeats, as determined by the XNU program of Claverie & States (Computers and Chemistry, 1993), or, for BLASTN, by the DUST program of Tatusov and Lipman (in preparation). Filtering can eliminate statistically significant but biologically uninteresting reports from the blast output (e.g., hits against common acidic-, basic- or proline-rich regions), leaving the more biologically interesting regions of the query sequence available for specific matching against database sequences.

Low complexity sequence found by a filter program is substituted using the letter "N" in nucleotide sequence (e.g., "NNNNNNNNNNNNN") and the letter "X" in protein sequences (e.g., "XXXXXXXXX"). Users may turn off filtering by using the "Filter" option on the "Advanced options for the BLAST server" page.

Filtering is only applied to the query sequence (or its translation products), not to database sequences. Default filtering is DUST for BLASTN, SEG for other programs.

It is not unusual for nothing at all to be masked by SEG, XNU, or both, when applied to sequences in SWISS-PROT, so filtering should not be expected to always yield an effect. Furthermore, in some cases, sequences are masked in their entirety, indicating that the statistical significance of any matches reported against the unfiltered query sequence should be suspect.

NCBI-gi

Causes NCBI gi identifiers to be shown in the output, in addition to the accession and/or locus name.

SEARCH STRATEGY

The fundamental unit of BLAST algorithm output is the High-scoring Segment Pair (HSP). An HSP consists of two sequence fragments of arbitrary but equal length whose alignment is locally maximal and for which the alignment score meets or exceeds a threshold or cutoff score. A set of HSPs is thus defined by two sequences, a scoring system, and a cutoff score; this set may be empty if the cutoff score is sufficiently high. In the programmatic implementations of the BLAST algorithm described here, each HSP consists of a segment from the query sequence and one from a database sequence. The sensitivity and speed of the programs can be adjusted via the standard BLAST algorithm parameters W, T, and X (Altschul et al., 1990); selectivity of the programs can be adjusted via the cutoff score.

A Maximal-scoring Segment Pair (MSP) is defined by two sequences and a scoring system and is the highest-scoring of all possible segment pairs that can be produced from the two sequences. The statistical methods of Karlin and Altschul (1990, 1993) are applicable to determining the significance of MSP scores in the limit of long sequences, under a random sequence model that assumes independent and identically distributed choices for the residues at each position in the sequences. In the programs described here, Karlin-Altschul statistics have been extrapolated to the task of assessing the significance of HSP scores obtained from comparisons of potentially short, biological sequences.

The approach to similarity searching taken by the BLAST programs is first to look for similar segments (HSPs) between

the query sequence and a database sequence, then to evaluate the statistical significance of any matches that were found, and finally to report only those matches that satisfy a user-selectable threshold of significance. Findings of multiple HSPs involving the query sequence and a single database sequence may be treated statistically in a variety of ways. By default the programs use "Sum" statistics (Karlin and Altschul, 1993). As such, the statistical significance ascribed to a set of HSPs may be higher than that ascribed to any individual member of the set. Only when the ascribed significance satisfies the user-selectable threshold (E parameter) will the match be reported to the user.

The task of finding HSPs begins with identifying short words of length W in the query sequence that either match or satisfy some positive-valued threshold score T when aligned with a word of the same length in a database sequence. T is referred to as the neighborhood word score threshold (Altschul et al., 1990). These initial neighborhood word hits act as seeds for initiating searches to find longer HSPs containing them. The word hits are extended in both directions along each sequence for as far as the cumulative alignment score can be increased. Extension of the word hits in each direction are halted when: the cumulative alignment score falls off by the quantity X from its maximum achieved value; the cumulative score goes to zero or below, due to the accumulation of one or more negative-scoring residue alignments; or the end of either sequence is reached.

KARLIN-ALTSCHUL STATISTICS

From Karlin and Altschul (1990), the principal equation relating the score of an HSP to its expected frequency of chance occurrence is:

$$E = K N \exp(-\text{Lambda } S)$$

where E is the expected frequency of chance occurrence of an HSP having score S (or one scoring higher); K and Lambda are Karlin-Altschul parameters; N is the product of the query and database sequence lengths, or the size of the search space; and exp is the exponentiation function.

Lambda may be thought of as the expected increase in reliability of an alignment associated with a unit increase in alignment score. Reliability in this case is expressed in units of information, such as bits or nats, with one nat being equivalent to 1/log(2) (roughly 1.44) bits.

The expectation E (range 0 to infinity) calculated for an alignment between the query sequence and a database sequence can be extrapolated to an expectation over the entire database search, by converting the pairwise expectation to a probability (range 0-1) and multiplying the result by the ratio of the entire database size (expressed in residues) to the length of the matching database sequence. In detail:

$$E\_database = (1 - \exp(-E))\ D / d$$

where D is the size of the database; d is the length of the

matching database sequence; and the quantity (1 - exp(-E)) is the probability, P, corresponding to the expectation E for the pairwise sequence comparison. Note that in the limit of infinite E, P approaches 1; and in the limit as E approaches 0, E and P approach equality. Due to inaccuracy in the statistical methods as they are applied in the BLAST programs, whenever E and P are less than about 0.05, the two values can be practically treated as being equal.

In contrast to the random sequence model used by Karlin-Altschul statistics, biological sequences are often short in length -- an HSP may involve a relatively large fraction of the query or database sequence, which reduces the effective size of the 2-dimensional search space defined by the two sequences. To obtain more accurate significance estimates, the BLAST programs compute effective lengths for the query and database sequences that are their real lengths minus the expected length of the HSP, where the expected length for an HSP is computed from its score. In no event is an effective length for the query or database sequence permitted to go below 1. Thus, the effective length of either the query or the database sequence is computed according to the following:

Length_eff = MAX(Length_real - Lambda S / H , 1)

where H is the relative entropy of the target and background residue frequencies (Karlin and Altschul, 1990), one of the statistics reported by the BLAST programs. H may be thought of as the information expected to be obtained from each pair of aligned residues in a real alignment that distinguishes the alignment from a random one.

SCORING SCHEMES

The default scoring matrix used by blastp, blastx, tblastn, and tblastx is the BLOSUM62 matrix (Henikoff and Henikoff, 1992).

Several PAM (point accepted mutations per 100 residues) amino acid scoring matrices are provided in the BLAST software distribution, including the PAM40, PAM120, and PAM250. While the BLOSUM62 matrix is a good general purpose scoring matrix and is the default matrix used by the BLAST programs, if one is restricted to using only PAM scoring matrices, then the PAM120 is recommended for general protein similarity searches (Altschul, 1991). The pam(1) program can be used to produce PAM matrices of any desired iteration from 2 to 511. Each matrix is most sensitive at finding similarities at its particular PAM distance. For more thorough searches, particularly when the mutational distance between potential homologs is unknown and the significance of their similarity may be only marginal, Altschul (1991, 1992) recommends performing at least three searches, one each with the PAM40, PAM120 and PAM250 matrices.

In blastn, the M parameter sets the reward score for a pair of matching residues; the N parameter sets the penalty score for mismatching residues. M and N must be positive and negative integers, respectively. The relative magnitudes of M and N determines the number of nucleic acid PAMs (point

accepted mutations per 100 residues) for which they are most sensitive at finding homologs. Higher ratios of M:N correspond to increasing nucleic acid PAMs (increased divergence). The default values for M and N, respectively 5 and -4, having a ratio of 1.25, correspond to about 47 nucleic acid PAMs, or about 58 amino acid PAMs; an M:N ratio of 1 corresponds to 30 nucleic acid PAMs or 38 amino acid PAMs. At higher than about 40 nucleic acid PAMs, or 50 amino acid PAMs, better sensitivity at detecting similarities between coding regions is expected by performing comparisons at the amino acid level (States et al., 1991), using conceptually translated nucleotide sequences (re: blastx, tblastn, and tblastx).

Independent of the values chosen for M and N, the default wordlength W=11 used by blastn restricts the program to finding sequences that share at least an 11-mer stretch of 100% identity with the query. Under the random sequence model, stretches of 11 consecutive matching residues are unlikely to occur merely by chance even between only moderately diverged homologs. Thus, blastn with its default parameter settings is poorly suited to finding anything but very similar sequences. If better sensitivity is needed, one should use a smaller value for W.

For the blastn program, it may be easy to see how multiplying both M and N by some large number will yield proportionally larger alignment scores with their statistical significance remaining unchanged. This scale-independence of the statistical significance estimates from blastn has its analog in the scoring matrices used by the other BLAST programs: multiplying all elements in a scoring matrix by an arbitrary factor will proportionally alter the alignment scores but will not alter their statistical significance (assuming numerical precision is maintained). From this it should be clear that raw alignment scores are meaningless without specific knowledge of the scoring matrix that was used.

SCORING REQUIREMENTS

Regardless of the scoring scheme employed, two stringent criteria must be met in order to be able to calculate the Karlin-Altschul parameters Lambda and K. First, given the residue composition for the query sequence and the residue composition assumed for the database, the alignment score expected for any randomly selected pair of residues (one from the query sequence and one from the database) must be negative. Second, given the sequence residue compositions and the scoring scheme, a positive score must be possible to achieve. For instance, the match reward score of blastn must have a positive value; and given the assumption made by blastn that the 4 nucleotides A, C, G and T are represented at equal 25% frequencies in the database, a wide range of value combinations for M and N are precluded from use -- namely those combinations where the magnitude of the ratio M:N is greater than or equal to 3.

GENETIC CODES

The parameter C can be set to a positive integer to select the genetic code that will be used by blastx and tblastx to translate the query sequence. The -dbgcode parameter can be used to select an alternate genetic code for translation of the database by the programs tblastn and tblastx. In each case, the default genetic code is the so-called "Standard" or "Universal" genetic code. To obtain a listing of the genetic codes available and their associated numerical identifiers, invoke blastx or tblastx with the command line parameter C=list. Note: the numerical identifiers used here for genetic codes parallel those defined in the NCBI software Toolbox; hence some numerical values will be skipped as genetic codes are updated.

The list of genetic codes available and their associated values for the parameters C and -dbgcode are:

1 Standard or Universal

2 Vertebrate Mitochondrial

3 Yeast Mitochondrial

4 Mold, Protozoan, Coelenterate Mitochondrial and Mycoplasma/Spiroplasma

5 Invertebrate Mitochondrial

6 Ciliate Macronuclear

9 Echinodermate Mitochondrial

10 Alternative Ciliate Macronuclear

11 Eubacterial

12 Alternative Yeast

13 Ascidian Mitochondrial

14 Flatworm Mitochondrial

P-VALUES, ALIGNMENT SCORES, AND INFORMATION

The Expect and P-values reported for HSPs are dependent on several factors including: the scoring system employed, the residue composition of the query sequence, an assumed residue composition for a typical database sequence, the length of the query sequence, and the total length of the database. HSP scores from different program invocations are appropriate for comparison even if the databases searched are of different lengths, as long as the other factors mentioned here do not vary. For example, alignment scores from searches with the default BLOSUM62 matrix should not be directly compared with scores obtained with the PAM120 matrix; and scores produced using two versions of the same PAM matrix, each created to different scales (see above), can not be meaningfully compared without conversion to the same scale.

Some isolation from the many factors involved in assessing the statistical significance of HSPs can be attained by observing the information content reported (in bits) for the alignments. While the information content of an HSP may change when different scoring systems are used (e.g., with different PAM matrices), the number of bits reported for an HSP will at least be independent of the scale to which the scoring matrix was generated. (In practice, this statement is not quite true, because the alignment scores used by the BLAST programs are integers that lack much precision). In other words, when conveying the statistical significance of an alignment, the alignment score itself is not useful unless the specific scoring matrix that was employed is also provided, but the informativeness of an alignment is a meaningful statistic that can be used to ascribe statistical significance (a P-value) to the match independently of specific knowledge about the scoring matrix.

SAMPLE OUTPUT

The BLAST programs all provide information in roughly the same format. First comes (A) an introduction to the program; (B) a histogram of expectations (see above) if one was requested; (C) a series of one-line descriptions of matching database sequences; (D) the actual sequence alignments; and finally the parameters and other statistics gathered during the search.

Sample blastp output from comparing pir|A01243|DXCH against the SWISS-PROT database is presented below.

A. Program Introduction

The introductory output provides the program name (BLASTP in this case), the version number (1.4.6MP in this case), the date the program source code last changed substantially (June 13, 1994), the date the program was built (Sept. 22, 1994), and a description of the query sequence and database to be searched. These may all be important pieces of information if a bug is suspected or if reproducibility of results is important.

The "Searching..." indicator indicates progress that the program made in searching the database. A complete database search will yield 50 periods (.), or one period per database sequence, whichever number is smaller. When searching a database consisting of 50 sequences or more, if fewer than 50 periods are displayed and the program aborted for some reason, dividing the number of periods by 0.5 will yield the approximate percentage (0-100%) of the database that was searched before the program died. If the program had difficulty making progress through the database, one or more asterisks (*) may be interspersed between the periods at one-minute intervals.

B. Histogram of Expectations

Shown in the output below is a histogram of the lowest (most significant) Expect values obtained with each database sequence. This information is useful in determining the numbers of database sequences that achieved a particular level of statistical significance. It indicates the number of database matches that would be reportable at various set-

tings for the expectation threshold (E parameter).

C. One-line Summaries

The one-line sequence descriptions and summaries of results are useful for identifying biologically interesting database matches and correlating this interest with the statistical significance estimates. Unless otherwise requested, the database sequences are sorted by increasing P-value (probability). Identifiers for the database sequences appear in the first column; then come brief descriptions of each sequence, which may need to be truncated in order to fit in the available space. The "High Score" column contains the score of the highest-scoring HSP found with each database sequence. The "P(N)" column contains the lowest P-value ascribed to any set of HSPs for each database sequence; and the "N" column displays the number of HSPs in the set which was ascribed the lowest P-value. The P-values are a function of N, as used in Karlin-Altschul "Sum" statistics or Poisson statistics, to treat situations where multiple HSPs are found. It should be noted that the highest-scoring HSP whose score is reported in the "High Score" column is not necessarily a member of the set of HSPs which yields the lowest P-value; the highest-scoring HSP may be excluded from this set on the basis of consistency rules governing the grouping of HSPs (see the -consistency option). Numbers of the form "7.7e-160" are in scientific notation. In this particular example, the number being represented is 7.7 times 10 to the minus 160th power. which is astronomically close to zero.

D. Alignments

Alignments found with the BLAST algorithm are ungapped. Several statistics are used to describe each HSP: the raw alignment Score; the raw score converted to bits of information by multiplying by Lambda (see the Statistics output); the number of times one might Expect to see such a match (or a better one) merely by chance; the P-value (probability in the range 0-1) of observing such a match; the number and fraction of total residues in the HSP which are identical; the number and fraction of residues for which the alignment scores have positive values. When Sum statistics have been used to calculate the Expect and P-values, the P-value is qualified with the word "Sum" and the N parameter used in the Sum statistics is provided in parentheses to indicate the number of HSPs in the set; when Poisson statistics have been used to calculate the Expect and P-values, the P-value is qualified with the word "Poisson". Between the two lines of Query and Subject (database) sequence is a line indicating the specific residues which are identical, as well as those which are non-identical but nevertheless have positive alignment scores defined in the scoring matrix that was used (the BLOSUM62 matrix in this case). Identical letters or residues, when paired with each other, are not highlighted if their alignment score is negative or zero. Examples of this would be an X juxtaposed with an X in two amino acid sequences, or an N juxtaposed with another N in two nucleotide sequences. Such ambiguous residue-residue pairings may be uninformative and thus lend no support to the overall alignment being either real or random; however, the informativeness of these pairings is left up to the user of the BLAST programs to decide, because any values desired can be specified in a scoring matrix of the user's own making.

BLASTP 1.4.6MP [13-Jun-94] [Build 13:58:36 Sep 22 1994]

Reference: Altschul, Stephen F., Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman (1990). Basic local alignment search tool. J. Mol. Biol. 215:403-10.

Query = pir|A01243|DXCH 232 Gene X protein - Chicken (fragment)
(232 letters)

Database: SWISS-PROT Release 29.0
38,303 sequences; 13,464,008 total letters.
Searching.....done

Observed Numbers of Database Sequences Satisfying
Various EXPECTation Thresholds (E parameter values)

Histogram units: = 31 Sequences : less than 31 sequences

```
EXPECTation Threshold
(E parameter)
   |
   V   Observed Counts-->
 10000 4863 1861 |============================================================
  6310 3002  782 |========================
  3980 2220  812 |=========================
  2510 1408  303 |=========
  1580 1105  393 |============
  1000  712  179 |=====
   631  533  161 |=====
   398  372   80 |==
   251  292   73 |==
   158  219   50 |=
   100  169   32 |=
  63.1  137   18 |:
  39.8  119    9 |:
  25.1  110    6 |:
  15.8  104    9 |:
>>>>>>>>>>>>>>>>>>>  Expect = 10.0, Observed = 95  <<<<<<<<<<<<<<<<<
  10.0   95    4 |:
  6.31   91    3 |:
  3.98   88    1 |:
  2.51   87    3 |:
  1.58   84    0 |
  1.00   84    2 |:
```

Sequences producing High-scoring Segment Pairs:	High Score	Smallest Sum Probability P(N)	N
sp|P01013|OVAX_CHICK GENE X PROTEIN (OVALBUMIN-RELATED) (...	1191	7.7e-160	1
sp|P01014|OVAY_CHICK GENE Y PROTEIN (OVALBUMIN-RELATED).	949	7.0e-127	1
sp|P01012|OVAL_CHICK OVALBUMIN (PLAKALBUMIN).	645	3.4e-100	2
sp|P19104|OVAL_COTJA OVALBUMIN.	626	1.2e-96	2
sp|P05619|ILEU_HORSE LEUKOCYTE ELASTASE INHIBITOR (LEI).	216	3.7e-71	3
sp|P80229|ILEU_PIG LEUKOCYTE ELASTASE INHIBITOR (LEI) (...	325	4.0e-71	2
sp|P29508|SCCA_HUMAN SQUAMOUS CELL CARCINOMA ANTIGEN (SCC...	439	3.5e-70	2
sp|P30740|ILEU_HUMAN LEUKOCYTE ELASTASE INHIBITOR (LEI) (...	211	1.3e-66	3
sp|P05120|PAI2_HUMAN PLASMINOGEN ACTIVATOR INHIBITOR-2, P...	176	1.8e-65	4

```
sp|P35237|PTI_HUMAN  PLACENTAL THROMBIN INHIBITOR.              473  1.3e-61   1
sp|P29524|PAI2_RAT   PLASMINOGEN ACTIVATOR INHIBITOR-2, T...   183  9.4e-61   4
sp|P12388|PAI2_MOUSE PLASMINOGEN ACTIVATOR INHIBITOR-2, M...   179  1.8e-60   4
sp|P36952|MASP_HUMAN MASPIN PRECURSOR.                         198  2.6e-58   4
sp|P32261|ANT3_MOUSE ANTITHROMBIN-III PRECURSOR (ATIII).       142  4.0e-48   5
sp|P01008|ANT3_HUMAN ANTITHROMBIN-III PRECURSOR (ATIII).       122  7.5e-48   5

WARNING:  Descriptions of 80 database sequences were not reported due to the
          limiting value of parameter V = 15.


  ... alignments with the top 8 database sequences deleted ...

>sp|P05120|PAI2_HUMAN PLASMINOGEN ACTIVATOR INHIBITOR-2, PLACENTAL (PAI-2)
           (MONOCYTE ARG- SERPIN).
           Length = 415

 Score = 176 (80.2 bits), Expect = 1.8e-65, Sum P(4) = 1.8e-65
 Identities = 38/89 (42%), Positives = 50/89 (56%)

Query:     1 QIKDLLVSSSTDLDTTLVLVNAIYFKGMWKTAFNAEDTREMPFHVTKQESKPVQMMCMNN 60
             +I +LL   S D DT +VLVNA+YFKG WKT F  +     PF V   +  PVQMM +
Sbjct:   180 KIPNLLPEGSVDGDTRMVLVNAVYFKGKWKTPFEKKLNGLYPFRVNSAQRTPVQMMYLRE 239

Query:    61 SFNVATLPAEKMKILELPFASGDLSMLVL 89
              N+  +   K +ILELP+A      L+L
Sbjct:   240 KLNIGYIEDLKAQILELPYAGDVSMFLLL 268

 Score = 165 (75.2 bits), Expect = 1.8e-65, Sum P(4) = 1.8e-65
 Identities = 33/78 (42%), Positives = 47/78 (60%)

Query:   155 ANLTGISSAESLKISQAVHGAFMELSEDGIEMAGSTGVIEDIKHSPESEQFRADHPFLFL 214
             AN +G+S    L +S+  H A ++++E+G E A  TG +   +      QF ADHPFLFL
Sbjct:   338 ANFSGMSERNDLFLSEVFHQAMVDVNEEGTEAAAGTGGVMTGRTGHGGPQFVADHPFLFL 397

Query:   215 IKHNPTNTIVYFGRYWSP 232
             I H  T  I++FGR+ SP
Sbjct:   398 IMHKITKCILFFGRFCSP 415

 Score = 144 (65.6 bits), Expect = 1.8e-65, Sum P(4) = 1.8e-65
 Identities = 26/62 (41%), Positives = 41/62 (66%)

Query:    90 LPDEVSDLERIEKTINFEKLTEWTNPNTMEKRRVKVYLPQMKIEEKYNLTSVLMALGMTD 149
             + D  + LE +E  I ++KL +WT+ + M +  V+VY+PQ K+EE Y L S+L ++GM D
Sbjct:   272 IADVSTGLELLESEITYDKLNKWTSKDKMAEDEVEVYIPQFKLEEHYELRSILRSMGMED 331

Query:   150 LF 151
              F
Sbjct:   332 AF 333

 Score = 61 (27.8 bits), Expect = 1.8e-65, Sum P(4) = 1.8e-65
 Identities = 10/17 (58%), Positives = 16/17 (94%)

Query:    81 SGDLSMLVLLPDEVSDL 97
             +GD+SM +LLPDE++D+
Sbjct:   259 AGDVSMFLLLPDEIADV 275


WARNING:  HSPs involving 86 database sequences were not reported due to the
          limiting value of parameter B = 9.
```

```
Parameters:
  V=15
  B=9
  H=1

  -ctxfactor=1.00
  E=10

  Query                          -----  As Used  -----     -----  Computed  ----
  Frame  MatID Matrix name       Lambda     K       H      Lambda     K       H
   +0      0   BLOSUM62          0.316   0.132   0.370     same     same    same

  Query
  Frame  MatID  Length  Eff.Length    E     S W   T  X     E2  S2
   +0      0      232       232      10.  57 3   11 22    0.22 33


Statistics:
  Query          Expected         Observed           HSPs       HSPs
  Frame  MatID  High Score       High Score       Reportable  Reported
   +0      0    62 (28.2 bits)  1191 (542.5 bits)     330        24

  Query         Neighborhd  Word      Excluded    Failed   Successful  Overlaps
  Frame  MatID   Words      Hits        Hits    Extensions Extensions  Excluded
   +0      0      4988     5661199     1146395    4504598    10187       13

  Database:  SWISS-PROT Release 29.0
    Release date:  June 1994
    Posted date:  1:29 PM EDT Jul 28, 1994
  # of letters in database:  13,464,008
  # of sequences in database:  38,303
  # of database sequences satisfying E:  95
  No. of states in DFA:  561 (55 KB)
  Total size of DFA:  110 KB (128 KB)
  Time to generate neighborhood:  0.03u 0.01s 0.04t  Real: 00:00:00
  No. of processors used:  8
  Time to search database:  32.27u 0.78s 33.05t  Real: 00:00:04
  Total cpu time:  32.33u 0.91s 33.24t  Real: 00:00:05

WARNINGS ISSUED:  2
```

COPYRIGHT

REFERENCES

Altschul, Stephen F. (1991). Amino acid substitution matrices from an information theoretic perspective. J. Mol. Biol. 219:555-65.

Altschul, S. F. (1993). A protein alignment scoring system sensitive at all evolutionary distances. J. Mol. Evol. 36:290-300.

Altschul, S. F., M. S. Boguski, W. Gish and J. C. Wootton (1994). Issues in searching molecular sequence databases. Nature Genetics 6:119-129.

Altschul, Stephen F., Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman (1990). Basic local alignment search tool. J. Mol. Biol. 215:403-10.

Claverie, J.-M. and D. J. States (1993). Information enhancement methods for large scale sequence analysis. Computers in Chemistry 17:191-201.

Gish, W. and D. J. States (1993). Identification of protein coding regions by database similarity search. Nature Genetics 3:266-72.

Henikoff, Steven and Jorga G. Henikoff (1992). Amino acid substitution matrices from protein blocks. Proc. Natl. Acad. Sci. USA 89:10915-19.

Karlin, Samuel and Stephen F. Altschul (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. Proc. Natl. Acad. Sci. USA 87:2264-68.

Karlin, Samuel and Stephen F. Altschul (1993). Applications and statistics for multiple high-scoring segments in molecular sequences. Proc. Natl. Acad. Sci. USA 90:5873-7.

States, D. J. and W. Gish (1994). Combined use of sequence similarity and codon bias for coding region identification. J. Comput. Biol. 1:39-50.

States, D. J., W. Gish and S. F. Altschul (1991). Improved sensitivity of nucleic acid database similarity searches using application specific scoring matrices. Methods: A companion to Methods in Enzymology 3:66-70.

Wootton, J. C. and S. Federhen (1993). Statistics of local complexity in amino acid sequences and sequence databases. Computers in Chemistry 17:149-163.

<TITLE> BLAST Reference Manual Pages </TITLE>
<!-- Changed by: Sergei Shavirin, 2-Apr-1996 -->
<BODY bgcolor="FFFFFF" link="0000FF" vlink="ff0000" text="000000" >
<h1 align = center>BLAST HELP MANUAL</h1>
<HR>

<h3>DESCRIPTION</h3>
<PRE>
This document describes the WWW BLAST interface.

BLAST (Basic Local Alignment Search Tool) is the heuristic search algorithm employed by the programs blastp, blastn, blastx, tblastn, and tblastx; these programs ascribe significance to their findings using the statistical methods of Karlin and Altschul (1990, 1993) with a few enhancements. The BLAST programs were tailored for sequence similarity searching -- for example to identify homologs to a query sequence. The programs are not generally useful for motif-style searching. For a discussion of basic issues in similarity searching of sequence databases, see Altschul et al. (1994).

The five BLAST programs described here perform the following tasks:

blastp compares an amino acid query sequence against a protein sequence database;

blastn compares a nucleotide query sequence against a nucleotide sequence database;

blastx compares the six-frame conceptual translation products of a nucleotide query sequence (both strands) against a protein sequence database;

tblastn compares a protein query sequence against a nucleotide sequence database dynamically translated in all six reading frames (both strands).

tblastx compares the six-frame translations of a nucleotide query sequence against the six-frame translations of a nucleotide sequence database.

<h1>BLAST Search parameters</h1>
<p>
<dl>
<dt>HISTOGRAM
<dd>Display a histogram of scores for each search; default is yes. (See parameter H in the BLAST Manual).
<dt>DESCRIPTIONS
<dd>Restricts the number of short descriptions of matching sequences reported to the number specified; default limit is 100 descriptions. (See parameter V in the manual page). See also EXPECT and CUTOFF.
<dt>ALIGNMENTS
<dd>Restricts database sequences to the number specified for

BLAST 2.0 RELEASE NOTES

(revised May 25, 1998)

Introduction

BLAST is a service of the National Center for Biotechnology Information (NCBI). A nucleotide or protein sequence sent to the BLAST server is compared against databases at the NCBI and a summary of matches is returned to the user.

The www BLAST server can be accessed through the home page of the NCBI at www.ncbi.nlm.nih.gov. Stand-alone BLAST binaries can be obtained from the NCBI FTP site. See the Stand-Alone Blast section for details.

The BLAST 2.0 release has significant differences from the BLAST 1.4 release. These include significant performance enhancements, the addition of 'gapping' routines, position-specific-iterated BLAST (see the PSI-Blast section) as well as extensive changes to the text report (see below), and the format of the databases (see the Stand-Alone Blast section). The options available and their command-line appearance have also changed substantially.

The BLAST 2.0 programs are described in a Nucleic Acids Research article. Please cite this reference if you publish the results of your BLAST query.

Blast Family of Programs

The BLAST family of programs allows all combinations of DNA or protein query sequences with searches against DNA or protein databases:

`blastp`	`compares an amino acid query sequence against a protein sequence database.`
`blastn`	`compares a nucleotide query sequence against a nucleotide sequence database.`
`blastx`	`compares the six-frame conceptual translation products of a nucleotide query sequence (both strands) against a protein sequence database.`
`tblastn`	`compares a protein query sequence against a`

```
            nucleotide     sequence    database    dynamically
            translated  in  all  six  reading   frames   (both
            strands).

    tblastx compares the six-frame translations of a nucleo-
            tide query sequence against the six-frame transla-
            tions of a nucleotide sequence database.
```

The default matrix for all protein-protein comparisons is BLOSUM62.

Gaps in Blast

Version 2.0 of BLAST allows the introduction of gaps (deletions and insertions) into alignments. With a gapped alignment tool, homologous domains do not have to be broken into several segments. Also, the scoring of gapped results tends to be more biologically meaningful than ungapped results.

The programs, blastn and blastp, offer fully gapped alignments. blastx and tblastn have 'in-frame' gapped alignments and use sum statistics to link alignments from different frames. tblastx provides only ungapped alignments.

Blast Query Format

The sequence sent to the BLAST server should be in FASTA format, described in http://www.ncbi.nlm.nih.gov/BLAST/fasta.html. A number of databases are also available. They are described in http://www.ncbi.nlm.nih.gov/BLAST/blast_databases.html.

Blast Report

The BLAST report consists of a number of sections. The descriptions below are for a blastp comparison, but the format for the other programs is analogous.

The BLAST report is not intended to be a parseable document. It is subject to change with little or no notice.

The BLAST report starts with some header information that lists the type of program (here blastp), the version (here 2.0.1), and a release date. Also listed are a reference to the BLAST program, the query definition line, and summary of the database used.

```
BLASTP 2.0.1 [Aug-20-1997]

Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997), "Gapped
BLAST and PSI-BLAST: a new generation of protein database search programs",
Nucleic Acids Res. 25:3389-3402.

Query= gi|129295|sp|P01013|OVAX_CHICK gene X protein - chicken (fragment)
         (232 letters)

Database: Non-redundant SwissProt sequences
           59,576 sequences; 21,219,450 total letters


One-line descriptions of the database matches found are presented next.  These
include a database sequence identifier, the corresponding definition line, as
well as the score (in bits) and the statistical significance ('E value') for this
match (please see the section on statistics for an explanation of bits and
```

significance). Consider the output below, from a gapped blastp comparison of SwissProt accession P01013 against the SwissProt database.

```
                                                                  High    E
Sequences producing significant alignments:                      Score  Value

sp|P01013|OVAX_CHICK GENE X PROTEIN (OVALBUMIN-RELATED)            442  e-124
sp|P01014|OVAY_CHICK GENE Y PROTEIN (OVALBUMIN-RELATED)            353  9e-98
sp|P01012|OVAL_CHICK OVALBUMIN (PLAKALBUMIN) (ALLERGEN GAL D II)   278  5e-75
sp|P19104|OVAL_COTJA OVALBUMIN                                     268  5e-72
sp|P48595|BOMA_HUMAN BOMAPIN (PROTEASE INHIBITOR 10)               199  2e-51
sp|P29508|SCC1_HUMAN SQUAMOUS CELL CARCINOMA ANTIGEN 1 (SCCA-1) ...  198  5e-51
sp|P80229|ILEU_PIG LEUKOCYTE ELASTASE INHIBITOR (LEI) (LEUCOCYTE...  197  1e-50
sp|P48594|SCC2_HUMAN SQUAMOUS CELL CARCINOMA ANTIGEN 2 (SCCA-2) ...  196  2e-50
sp|P50453|PTI9_HUMAN CYTOPLASMIC ANTIPROTEINASE 3 (CAP3) (PROTEA...  195  6e-50
sp|P05619|ILEU_HORSE LEUKOCYTE ELASTASE INHIBITOR (LEI)            193  2e-49
```

The first match, in this case, is the actual query sequence. The identifiers shown here are all from SwissProt, so they all have 'sp' in the first field, followed by the accession, and then a Locus name. The syntax of these identifiers is discussed in more detail in the appendices of ftp://ncbi.nlm.nih.gov/blast/db/README The definition lines are taken from the definition line in the database, with the ellipsis (e.g., P29508) indicating that the definition line was too long to for the space available.

Ungapped alignments and results from blastx and tblastn will have an additional column ('N'), displaying the number of different segment pairs used to produce the alignment, according to the Karlin-Altschul statistics.

Each alignment is preceded by the sequence identifier, the full definition line and the length of the database sequence. Next come the score (in bits as well as the raw score) as well as the statistical significance of the match, followed by the number of identities and positive matches according to the scoring system (e.g., BLOSUM62) and, if applicable, the number of gaps in the alignment. Finally the actual alignment is shown, with the query on top and the database match labeled as 'Sbjct'. Between the two sequences the residue is shown if it is conserved, a '+' is shown if there is a positive match. One or more dashes, '-', indicates insertions or deletions. The example below is the third sequence listed in the one-line descriptions above.

```
>sp|P01012|OVAL_CHICK OVALBUMIN (PLAKALBUMIN) (ALLERGEN GAL D II)
          Length = 386

Score =  278 bits (744), Expect = 5e-75
Identities = 149/231 (64%), Positives = 182/231 (78%), Gaps = 2/231 (0%)

Query 2   IKDLLVSSSTDLDTTLVLVNAIYFKGMWKTAFNAEDTREMPFHVTKQESKPVQMMCMNNS 61
          I+++L  SS D  T +VLVNAI FKG+W+ AF  EDT+ MPF VT+QESKPVQMM
Sbjct 158 IRNVLQPSSVDSQTAMVLVNAIVFKGLWEKAFKDEDTQAMPFRVTEQESKPVQMMYQIGL 217

Query 62  FNVATLPAEKMKILELPFASGDLSMLVLLPDEVSDLERIEKTINFEKLTEWTNPNTMEKR 121
          F VA++ +EKMKILELPFASG +SMLVLLPDEVS LE++E  INFEKLTEWT+ N ME+R
Sbjct 218 FRVASMASEKMKILELPFASGTMSMLVLLPDEVSGLEQLESIINFEKLTEWTSSNVMEER 277

Query 122 RVKVYLPQMKIEEKYNLTSVLMALGMTDLFIPSANLTGISSAESLKISQAVHGAFMELSE 181
          ++KVYLP+MK+EEKYNLTSVLMA+G+TD+F  SANL+GISSAESLKISQAVH A  E++E
Sbjct 278 KIKVYLPRMKMEEKYNLTSVLMAMGITDVFSSSANLSGISSAESLKISQAVHAAHAEINE 337

Query 182 DGIEMAGSTGVIEDIKHSPESEQFRADHPFLFLIKHNPTNTIVYFGRYWSP 232
           G E+ GS      +  +  SE+FRADHPFLF IKH  TN +++FGR  SP
Sbjct 338 AGREVVGSAEA--GVDAASVSEEFRADHPFLFCIKHIATNAVLFFGRCVSP 386
```

The last section lists specifics about the database searched as well as statistical and search parameters used:

```
  Database: Non-redundant SwissProt sequences
    Posted date:  Aug 14, 1997  9:52 AM
  Number of letters in database: 21,219,450
  Number of sequences in database:  59,576

Lambda      K      H
   0.317    0.132    0.377

Gapped
Lambda      K      H
   0.255   0.0350    0.190


Matrix: BLOSUM62
Gap Penalties: Existence: 10, Extension: 1
Number of Hits to DB: 8938654
Number of Sequences: 59576
Number of extensions: 335248
Number of successful extensions: 1188
Number of sequences better than 10: 116
Number of HSP's better than 10.0 without gapping: 106
Number of HSP's successfully gapped in prelim test: 10
Number of HSP's that attempted gapping in prelim test: 868
Number of HSP's gapped (non-prelim): 120
length of query: 232
length of database: 21219450
effective HSP length: 52
effective length of query: 180
effective length of database: 18121498
effective search space: -1033097656
T: 11
A: 40
X1: 16 ( 7.3 bits)
X2: 40 (14.7 bits)
X3: 67 (24.6 bits)
S1: 41 (21.7 bits)
S2: 64 (28.4 bits)
```

Blast Statistics and Scores

One may judge the results of a blast search by two numbers. One is the 'bit' score, which is defined as:

```
S' (bits) =  [lambda * S (raw) - ln K] / ln 2
```

where lambda and K are Karlin-Altschul parameters. The expression of the score in terms of bits makes it independent of the scoring system used (i.e., which matrix). The Expect value estimates the statistical significance of the match, specifying the number of matches, with a given score, that are expected in a search of a database of this size absolutely by chance. An Expect value of two, with a given score, would indicate that two matches with this score, are expected purely by chance. The expect value changes with the size of the database (in a larger database more chance matches with a given score are expected) and is the most intuitive way to rank results or compare the results of one query run against two different databases.

Stand-Alone Blast

This section is only applicable if a users wishes to run stand-alone BLAST at their own institution. One reason to do so might be the wish to use private databases not available at the NCBI.

Users of www or network BLAST do not need to read these sections.

BLAST binaries are provided for IRIX6.2, Solaris2.5, DEC OSF1 (ver. 4), and Win32 systems. We will attempt to produce binaries for other platforms upon request.

Stand-alone binaries are available from ftp://ncbi.nlm.nih.gov/blast/executables.

The source code for BLAST 2.0 is part of the NCBI toolkit. The NCBI toolkit may be obtained from ftp://ncbi.nlm.nih.gov/toolbox/ncbi_tools. Use the makedemo Makefiles to build blastpgp, blastall, and formatdb after compiling the rest of the toolkit (see Compiling Blast).

Please remember to FTP in binary mode.

Formatdb

Formatdb, should be used to format the FASTA databases for both protein and DNA databases for BLAST 2.0. This must be done before blastall or blastpgp can be run locally. The format of the databases has been changed substantially from the BLAST 1.4 release. A major improvement in this format over the old one is that ambiguity information for DNA sequences is now retrieved from the files produced by formatdb, rather than from the original FASTA file. The original FASTA file is no longer needed for the BLAST runs. Formatdb may be obtained with the other BLAST binaries from the executables directory (see above). The input for formatdb may be either ASN.1 or FASTA. Use of ASN.1 is advantageous for those sites that might also wish to format the ASN.1 in different ways, such as a GenBank report. Usage of formatdb may be obtained by executing formatdb and a dash:

```
formatdb   arguments:

  -t  Title for database file [String]  Optional
  -i  Input file for formatting (this parameter must be set) [File In]
  -l  Logfile name: [File Out]  Optional
    default = formatdb.log
  -p  Type of file
         T - protein
         F - nucleotide [T/F]  Optional
    default = T
  -o  Parse options
         T - True: Parse SeqId and create indexes.
         F - False: Do not parse SeqId. Do not create indexes.
 [T/F]  Optional
    default = F
  -a  Input file is database in ASN.1 format (otherwise FASTA is expected)
         T - True,
         F - False.
 [T/F]  Optional
    default = F
  -b  ASN.1 database in binary mode
         T - binary,
         F - text mode.
 [T/F]  Optional
    default = F
  -e  Input is a Seq-entry [T/F]  Optional
    default = F
```

The "-p" option has two different meaning depending on whether input database is in FASTA or ASN.1 format. In case of FASTA, the "-p" specifies type of input database. In case of ASN.1, the option specifies the type of

sequence to be indexed for BLAST.

If the "-o" option is TRUE (and the input database is in FASTA format), then the database identifiers in the FASTA definition line must follow the convention described in the appendices of ftp://ncbi.nlm.nih.gov/blast/db/README

It is always advantageous to use the '-o' option if the database identifiers are in the format specified above. If the database identifiers are in the parseable formatdb produces additional indices allowing retrieval from the databases by identifier. The databases on the NCBI FTP site contain parseable identifiers. It is sufficient if the first word on the FASTA defintion line is a unique identifier (e.g., ">3091 Alcoho de..."). It is necessary to use parseable identifiers for the following cases:

```
1.) If ASN.1 is to be produced from blastall or blastpgp, then "-o" must be TRUE.

2.) master-slave alignments are desired (i.e., the '-m' option with a non-zero value is us

3.) The gi's are desired as part of the output (i.e., '-I' is used).

4.) fastacmd is used to fetch sequences from the database by accession or gi.
```

An input ASN.1 database may be represented in two formats - ascii text and binary. The "-b" option, if TRUE, specifies that input ASN.1 database is in binary format. The option is ignored in case of FASTA input database.

An input ASN.1 database (either text ascii or binary) may contains Bioseq-set or just one Bioseq. In the latter case the "-e" switch should be set to TRUE.

Blastall

Blastall may be used to perform all five flavors of blast comparison. One may obtain the blastall options by executing 'blastall -' (note the dash). A typical blastall to perform a blastn search (nucl. vs. nucl.) of a file called QUERY would be:

```
blastall -p blastn -d nr -i QUERY -o out.QUERY

The output is placed into the output file out.QUERY and the search is performed
against the 'nr' database.  If a protein vs. protein search is desired,
then 'blastn' should be replaced with 'blastp' etc.


Some of the most commonly used blastall options are:

blastall   arguments:

  -p  Program Name [String]

        Input should be one of "blastp", "blastn", "blastx", "tblastn", or "tblastx".

  -d  Database [String]
    default = nr

        Version 2.0.4 and higher will accept multiple database names (bracketed by quotati
        An example would be

                -d "nr est"

        which will search both the nr and est databases, presenting the results as if one
```

```
        'virtual' database consisting of all the entries from both were searched.   The
        statistics are based on the 'virtual' database.


  -i  Query File [File In]
    default = stdin

        The query should be in FASTA format.  If multiple FASTA entries are in the input
        file, all queries will be searched.

  -e  Expectation value (E) [Real]
    default = 10.0

  -o  BLAST report Output File [File Out]  Optional
    default = stdout

  -F  Filter query sequence (DUST with blastn, SEG with others) [T/F]
    default = T

        See the "Low-complexity Filters" section below for details.
```

Blastpgp

Blastpgp performs gapped blastp searches and can be used to perform iterative searches in psi-blast mode. See the PSI-Blast section for a description of this binary. The options may be obtained by executing 'blastpgp -'.

Software requirements

Blast 2.0 uses threads to perform multi-processing searches. OS requirements on SGI's are IRIX 6 (with relevant threads patches, see below), any Solaris version, or a version of DEC UNIX. IRIX 5 may be used if multi-processing is not enabled.

```
SGI recommends the following threads patches on IRIX6 systems:

   For 6.2 systems, install SG0001404, SG0001645, SG0002000, SG0002420 and SG0002458 (in t
   For 6.3 systems, install SG0001645, SG0002420 and SG0002458 (in that order)
   For 6.4 systems, install SG0002194, SG0002420 and SG0002458 (in that order)

These patches can be obtained by calling SGI customer service or from the web: http://supp
```

System recommendations

BLAST uses memory-mapped files (on UNIX and NT systems), so it runs best if it can read the entire BLAST database into memory, then keep on using it there. Resources consumed reading a database into memory can easily outweight the cost of a BLAST search, so that the memory of a machine is normally more important than the CPU speed. This means that one should have sufficient memory for the largest BLAST database one will use, then run all the searches against this databases in serial, then run queries against another database in serial. This guarantees that the database will be read into memory only once. As of this date (Aug. 1997) the EST FASTA file is about 500 Meg, which translates to about 170-200 Meg of BLAST database. At least another 100-200 Meg should be allowed for memory consumed by the actual BLAST program. All of the FASTA databases together are about 1.5 Gig, the BLAST databases produced from this will probably be about another Gig or so. 4 Gig of disk space, to make room for software and output, is probably a pretty good bet.

Setup

BLAST needs to know where the NCBI data is. This is specified by the main configuration file for the NCBI toolkit (".ncbirc" on UNIX systems, ncbi.ini on Windows, analogous names on other platforms). If BLAST is the ONLY NCBI application that will be used, it is sufficient to have the following two-line configuration file:

```
[NCBI]
   Data=/am/ncbiapdata/data
```

BLAST looks for the file 'seqcode.val', 'gc.code', and 'BLOSUM62' in the "Data" directory (e.g., "/am/ncbiapdata/data/seqcode.val"). A directory different than "/am/ncbiapdata/data" can be used if this is desired. The files seqcode.val, gc.val, and BLOSUM62 can be found in the data directory of the toolbox (i.e., ncbi/data). The .ncbirc should be either in the directory from which BLAST is called, the user's home directory, or in the directory set by the environment variable "NCBI".

Database and matrix directories

On UNIX systems environment variables can be setenv to specify the directory of the database (BLASTDB) and matrices (BLASTMAT).

On non-UNIX systems it is currently necessary to run BLAST from the same directory as the databases, or explicitly write out the path. BLAST will soon read the NCBI configuration file for database directory information.

Low-complexity Filters

BLAST 2.0 uses the dust low-complexity filter for blastn and seg for the other programs. 'dust' is an integral part of the NCBI toolkit and is accessed automatically. 'seg' is a stand-alone program written only for UNIX. It may be obtained from ftp://ncbi.nlm.nih.gov/pub/seg/seg/. The environment variable for filters is BLASTFILTER.

BLAST databases

The FASTA files used by the NCBI to produce BLAST databases are available on the NCBI FTP site in ftp://ncbi.nlm.nih.gov/blast/db/. Please see the README for details.

Compiling Blast

```
This section provides abbreviated instructions on building BLAST
for some popular platforms.  It also provides guidance on how to
build the toolkit in a threaded manner, so that multi-threaded
BLAST may be run.  It is still recommended that the README provided
with the NCBI toolkit be referred to.

To make BLAST it is first necessary to make the standard NCBI libraries (this
actually contains most of the BLAST source code).  It is then necessary to
compile the demo's, which contains blastall, blastpgp, and formatdb.  BLAST
does not require the network or vibrant libraries.

================================================================================
                Solaris 2.5
================================================================================

1.) Obtain the toolkit archive from the NCBI FTP site, download in binary mode,
uncompress, and untar.

2.) cd ncbi/build
```

```
3.) cp ../make/*.unx .

4.) mv makeall.unx makefile

5.) make with:
        make LCL=sol CC=gcc OTHERLIBS="-lm -lthread"

6.) Make demos with:
        make -f makedemo.unx CC=gcc OTHERLIBS="-lm -lthread" THREAD_OBJ="ncbithr.o"
```

```
================================================================================
                IRIX6
================================================================================

1.) Obtain the toolkit archive from the NCBI FTP site, download in binary mode,
uncompress, and untar.

2.) cd ncbi/build

3.) cp ../make/*.unx .

4.) mv makeall.unx makefile

5.) Make with:
        make LCL=sgi OTHERLIBS="-lm -lPW -lpthread" CFLAGS1="-c -O -DPOSIX_THREADS_AVAIL"

6.) Make demos with:
                make -f makedemo.unx LCL=sgi OTHERLIBS="-lm -lPW -lpthread" THREAD_OBJ="nc
```

```
================================================================================
                DEC Alpha (v. 4.0)
================================================================================

1.) Obtain the toolkit archive from the NCBI FTP site, download in binary mode,
uncompress, and untar.

2.) cd ncbi/build

3.) cp ../make/*.unx .

4.) mv makeall.unx makefile

5.) Make with:
        make LCL=alf CC=cc RAN=ranlib OTHERLIBS="-lm -pthread"

6.) Make demos with:
                make -f makedemo.unx LCL=alf CC=cc RAN=ranlib OTHERLIBS="-lm -pthread" THR
```

```
================================================================================
                LINUX
================================================================================

1.) Obtain the toolkit archive from the NCBI FTP site, download in binary mode,
```

```
uncompress, and untar.

2.) cd ncbi/build

3.) cp ../make/*.unx .

4.) mv makeall.unx makefile

5.) Make with:
        make LCL=lnx CC=gcc RAN=ranlib OTHERLIBS="-lm -lpthread"

6.) Make demos with:
        make -f makedemo.unx LCL=lnx CC=gcc RAN=ranlib OTHERLIBS="-lm -lpthread" THREAD_OB
```

Database Format

The format of the BLAST databases has changed for the 2.0 release and is not compatiable with the databases used in the 1.4 release. The change was made to eliminate an unpleasant feature of the 1.4 databases: ambiguity information for nucleotide sequences was not stored in the compressed file, but rather the original FASTA file had to be accessed for this information. This leads to significant slow-downs in BLAST comparisons for databases, such as dbest, that contain a large number of ambiguity characters.

PSI-Blast

```
The blastpgp program can do an iterative search in which
sequences found in one round of searching are used to build
a score model for the next round of searching. In this usage,
the program is called Position-Specific Iterated BLAST, or PSI-BLAST.
As explained in the accompanying paper, the BLAST algorithm is
not tied to a specific score matrix. Traditionally, it has been
implemented using an AxA substitution matrix where A is the alphabet size.
PSI-BLAST instead uses a QxA matrix, where Q is the length of the query
sequence; at each position the cost of a letter depends on the position
w.r.t. the query and the letter in the subject sequence.

The position-specific matrix for round i+1 is built from a constrained
multiple alignment among the query and the sequences found with
sufficiently low e-value in round i.  The top part of the output for
each round distinguishes the sequences into: sequences found
previously and used in the score model, and sequences not used in the
score model. The output currently includes lots of diagnostics
requested by users at NCBI. To skip quickly from the output of
one round to the next, search for the string "producing", which is
part of the header for each round and likely does not appear elsewhere
in the output. PSI-BLAST "converges" and stops if all sequences
found at round i+1 below the e-value threshold were already in
the model at the beginning of the round.

There are several blastpgp parameters specifically for PSI-BLAST:
-j   is the maximum number of rounds (default 1; i.e., regular BLAST)
-e   is the e-value threshold for including sequences in the
     score matrix model (default 0.01)
-c   is the "constant" used in the pseudocount formula specified in the
     paper (default 10)

The -C and -R flags provide a "checkpointing" facility whereby
a score model can be stored and later reused.
   -C  stores the query and frequency count ratio matrix in a
```

file
-R restarts from a file stored previously.
When using -R, it is required that the query specified on the command line match exactly the query in the restart file.
The checkpoint files are stored in a byte-encoded (not human readable) format, so as to prevent roundoff error between writing and reading the checkpoint.
Users who also develop their own sequence analysis software may wish to develop their own scoring systems. For this purpose the code in posit.c that writes out the checkpoint can be easily adapated to write out scoring systems derived by other algorithms in such a way that PSI-BLAST can read the files in later.
The checkpoint structure is general in the sense that it can handle any position-specific matrix that fits in the Karlin-Altschul statistical framework for BLAST scoring.

References

Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997), "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs", Nucleic Acids Res. 25:3389-3402.

Karlin, Samuel and Stephen F. Altschul (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. Proc. Natl. Acad. Sci. USA 87:2264-68.

Karlin, Samuel and Stephen F. Altschul (1993). Applications and statistics for multiple high-scoring segments in molecular sequences. Proc. Natl. Acad. Sci. USA 90:5873-7.

Release History

Notes for 2.0.5 release:

Enhancements:

1.) The BLAST version is printed by formatdb in it's log file.

2.) Multi-database searches no longer require that the -o option be used when preparing the databases (i.e., with formatdb).

Bugs fixed:

1.) A serious bug with multi-database iterative searches was fixed (thanks to Steve Brenner for providing an example).

2.) 'lcl' is not formatted in the BLAST report when the sequence identifier is a local identifier or does not contain a bar ("|").

3.) A large memory leak in formatdb was fixed.

4.) An unnecessary cast that caused formatdb to fail on Solaris 2.5 machines if the binary was made under 2.6 was fixed.

5.) Better error checking was added to protect against core-dumps.

6.) Some problems with the sum statistics treatment of the blastx and tblastn programs reported by D. Rozenbaum were fixed. The number of alignments involved in a sum group was misrepresented. Also the incorrect length for the database sequence was used, sometimes casuing a slight change in the value reported.

7.) A problem with blastpgp was fixed that reported incorrect values for matrices other than BLOSUM62 during iterative searches.

Notes for 2.0.4 release:

Enhancements:

1.) multiple database searches:

Version 2.0.4 will accept multiple database names (bracketed by quotations). An example would be .

```
-d "nr est"
```

which will search both the nr and est databases, presenting the results as if one 'virtual' database consisting of all the entries from both were searched. The statistics are based on the 'virtual' database.

2.) new options:

```
-W  Word size, default if zero [Integer]
  default = 0
-z  Effective length of the database (use zero for the real size) [Integer]
  default = 0
```

3.) The number of identities, positives, and gaps are now printed out before the alignments for gapped blastx, tblastn, and tblastx. Additionally this feature is now also enabled for ungapped BLAST.

4.) Formatdb now accepts ASN.1, as well as FASTA, as input.

Bugs fixed:

1.) In blastx, tblastn, and tblastx a codon was incorrectly formatted as a start codon in some cases.

2.) The last alignment of the last sequence being presented was incorrectly dropped in some cases. This change could affect the statistical significance of the last database sequence if the dropped alignment had a lower e-value than any other alignments from the same database sequence.

If you have problems or comments...

Back to PBIL home page

BIOINFORMATICS

Bioinformatics
Vol. 14 no. 4 1998
Pages 349-356

Searching DNA databases for similarities to DNA sequences: when is a match significant?

Isobel Anderson and Andy Brass*

School of Biological Sciences, University of Manchester, 2.205 Stopford Building, Oxford Road, Manchester M13 9PT, UK

Received on October 31, 1997; revised on December 22, 1997; accepted on January 5, 1998

Abstract

Motivation: *Searching DNA sequences against a DNA database is an essential element of sequence analysis. However, few systematic studies have been carried out to determine when a match between two DNA sequences has biological significance and this is limiting the use that can be made of DNA searching algorithms.*

Results: *A test set of DNA sequences has been constructed consisting of artificially evolved and real sequences. This set has been used to test various database searching algorithms (BLAST, BLAST2, FASTA and Smith–Waterman) on a subset of the EMBL database. The results of this analysis have been used to determine the sensitivity and coverage of all of the algorithms. Guidelines have been produced which can be used to assess the significance of DNA database search results. The Smith–Waterman algorithm was shown to have the best coverage, but the worst sensitivity, whereas the default BLASTN algorithm (word length set to 11) was shown to have good sensitivity, but poor coverage. A sensible compromise between speed, sensitivity and coverage can be obtained using either the FASTA or BLAST (word length set to 6) algorithms. However, analysis of the results also showed that no algorithm works well when the length of the probe sequence is <200 bases. In general, matches can accurately be identified between coding regions of DNA sequences when there is >35% sequence identity between the corresponding proteins. Searching a DNA sequence against a DNA sequence database can, therefore, be a useful tool in sequence analysis.*

Availability: *The test sets used are available via anonymous ftp from mbisg2.sbc.man.ac.uk in the directory /pub/cabios/testdata/*

Contact: *I.Anderson@stud.man.ac.uk; abrass@man.ac.uk*

*To whom correspondence should be addressed

Introduction

Sequence comparison is a key tool in bioinformatics analysis. One of the best ways of assigning a putative function to a given sequence is to compare it to sequences of known function. There are, therefore, many occasions when a newly determined DNA sequence must be compared against sequence databases.

If a DNA sequence contains a protein coding region, then sequence comparison at the protein level is normally preferred. There are a number of reasons for this, such as:

1. Searches at the protein level should be more sensitive. There is degeneracy at the DNA level. Different codons can encode the same amino acid. This means that two identical protein sequences can differ greatly at the DNA level.

2. We have a better understanding of what constitutes a significant protein–protein hit. A number of papers have considered the problem of comparing protein sequences against protein sequence databases. Sander and Schneider (1991) showed that there must be 25% sequence identity over >80 residues to be confident of topological similarity.

3. Protein sequences are shorter than the corresponding DNA sequences. Protein sequence databases are smaller than DNA databases. The latest release of Swissprot contains 59 021 entries (October 1996). The main DNA sequence databases, EMBL, GenBank and DDBJ, contain >1 432 941 sequences (June 1997). Protein database are therefore much quicker to search.

Therefore, in the majority of cases, sequence comparisons of DNA sequences containing a protein coding region are run using a program such as BLASTX which translates the DNA sequence into all possible reading frames and then compares the resultant protein sequences against protein sequence databases.

There are a number of occasions when this approach might not give the best results, or when additional information can be obtained by comparing the raw DNA sequence against a DNA sequence database. For example:

1. The primary repository of sequence data is DNA databases. Therefore, DNA databases contain the most up-to-date sequences. Annotation bottlenecks can mean that there is a delay in the coding regions of DNA sequences being deposited in protein sequence databases.

2. The protein databases may contain false translations of raw DNA sequences (Bork and Bairoch, 1996).

3. The query sequence may be non-coding, or, especially in the case of error-prone expressed sequence tags, the correct open reading frame may be difficult to determine (Mann, 1996).

In such circumstances, running a DNA database search might be a sensible option. However, although the techniques for assigning significance for running protein sequence searches against the databases have been well studied, very little work has been done on assessing searches of DNA sequences against DNA sequence databases. Three questions in particular need to be addressed. (i) What are the best algorithms for comparing DNA sequences against a DNA sequence database? (ii) Is it possible to tell from the alignment scores when a hit is significant? (iii) What error rate is expected from the search, i.e. what sort of false-positive/false-negative rates should be expected for the searches?

In this paper, we have therefore generated a model system in which to explore the performance of algorithms which compare DNA sequences against DNA sequence databases. The aims of the work are 2-fold: firstly, to assess the strengths and weaknesses of existing search algorithms, and secondly to provide researchers with a set of guidelines by which to assess the significance of DNA search results.

Systems and methods

Creation of test sets

Test sets for analysing the performance of protein sequence database searches have been created (Shpaer *et al.*, 1996) or can be derived from protein classification systems such as SCOP (Murzin *et al.*, 1995). No such test sets are available for testing DNA database searches. For this work, we have therefore constructed an extensive, carefully controlled test data set in which the sequences to be compared against the database have either been artificially generated or carefully chosen from real sequence families.

Consider choosing some DNA sequence, $S(0)$, from a DNA sequence database, D, where $S(0)$ is reasonably long and does not consist entirely of low-complexity sequences. A sequence comparison algorithm run using $S(0)$ as a probe against the database should find that the best match to $S(0)$ is $S(0)$ itself. Now consider modifying the sequence $S(0)$, using some form of evolutionary algorithm, to produce a new sequence at a distance d away from the original sequence, $S(d)$, where d is measured using some form of sequence space metric. Consider now using the sequence $S(d)$ as a probe to search the database D. If d is small, we would expect that $S(d)$ would find the original sequence $S(0)$ as one of its top matches. However, for large values of d, we would have to move so far away from $S(0)$ that it is no longer recognized as a homologous sequence. In principle, the more sensitive the sequence comparison algorithm, the larger the value of d that can be used, such that the sequence $S(d)$ finds a significant match against $S(0)$. We can therefore explore the behaviour of different comparison algorithms by exploring how effectively they can locate homologous matches to a range of different seed sequences, $S(0)$, as a function of the distance that the probe sequence, $S(d)$, has been evolved.

For this technique to work, we therefore need an algorithm to evolve DNA sequences and a metric to measure distance in sequence space. Sequence evolution has been studied extensively for regions of DNA that code for protein. Models of DNA evolution are much less well understood in other regions of DNA. In this study, we have therefore concentrated on testing DNA sequence searching for protein coding regions in DNA. Many metrics have been proposed for such applications, for this study we are using PAM (point accepted mutations) distance (Dayhoff *et al.*, 1978). PAM distance is the number of substitutions that occur per 100 amino acids as one protein sequence evolves to another, allowing for the fact that multiple substitutions will have occurred at some positions. The PAM matrix gives the log-odds probability of one amino acid being substituted for another based upon mutational data. We are using the PAM distance between the corresponding aligned protein sequences as a measure of distance between two DNA sequences.

To create the test sets of artificially evolved sequences, 'seed' sequences were mutated using the Evolve algorithm (Slater, 1995). This algorithm simulates evolution in molecular sequences, with mutation taking place at the DNA level, and selection at the protein level, until a specified PAM distance between the original and evolved protein sequences has been reached. It is worth noting that insertions and deletions (indels) are added to the sequence. The gap length is determined by means of a Zipfian distribution (Benner *et al.*, 1993), and is usually zero. There is an equal chance of the indel being an insertion or a deletion. When a deletion is selected, the corresponding region of the DNA sequence is deleted. When an insertion is selected, the composition of the DNA sequence to be added is determined using a codon usage table to weight the decisions as to which codons to use.

In all cases, the searches were performed against a database containing real sequences, the primate subset of EMBL which contains around 60 000 sequences. DNA databases do not contain sequences randomly distributed through sequence space. For example, the current data contained within EMBL database are biased towards genes that are or have

been of scientific interest. It is possible to show that the chance of finding a match to an unrelated sequence in a database (i.e. a false-positive result) is proportional to $\sqrt{\ln(N)}$, where N is the number of sequences in the databases (I.Anderson, unpublished results), i.e. only very weakly dependent on the size of the database. The primate subset of EMBL is, therefore, a convenient subset to use to provide representative results at a relatively modest computational cost.

The first test set was constructed from a number of 'seed' sequences, shown in Table 1. The seed sequences were chosen to reflect different protein structures and different lengths of sequence so as to remove as far as possible bias in the test set. Three sequences were chosen according to secondary structure classification of the corresponding proteins (either all α helices, all β sheets, or both α helices and β sheets) using SCOP as a reference (Murzin *et al.*, 1995; http://www.bio.cam.ac.uk/scop). Another seven sequences were chosen to have lengths varying from 100 to 1000 bp. RepeatMasker (http://ftp.genome.washington.edu/) was used to ensure that the seed sequences were free from interspersed repeats and low-complexity regions. The Evolve program was used to generate randomly a set of 10 new sequences at a specific PAM distance from each of the seed sequences. Sets of sequences were generated at distances of 25, 50, 75, 100, 125, 150, 175, 200, 225 and 250 PAMs from each of the seed sequences. In total, there were therefore 1000 sequences in the artificially created test set, 100 sequences being generated from each of the 10 seed sequences.

Table 1. Genes included in the test set

Gene name	Accession no.	ORF length	Classification
Ciliary neurotrophic factor	x60542	597	All α
Tyrosine phosphatase	u02681	633	α/β
Retinol binding protein	x00129	594	All β
Human paired-box-protein PAX8	S77904	99	Length 100
Human adenosine deaminase (ADA) gene	X02195	146	Length 150
Epsilon globin gene	U11712	210	Length 200
Pro-alpha-2 chain of type I procollagen	V00503	402	Length 400
AQP3 gene	D25280	606	Length 600
cAMP responsive element binding protein γ subunit	L05913	811	Length 800
Progesterone receptor-associated p48 protein	U28918	1005	Length 1000

Table 2. Summary of database search methods and parameters used

Database search method	Database searched	Parameters	Scoring matrix	Type of query sequence test set used	Average search time (s)
Smith–Waterman	PRI	gap = –4.5, –0.05	Match = +1 Mismatch = –0.6	Gene type Gene length	720
BLASTN (default)	PRI	Word size = 11	Match = +5 Mismatch = –4	Gene type Gene length	34
BLASTN (w = 6)	PRI	Word size = 6	Match = +5 Mismatch = –4	Gene length	347
FASTA (default)	PRI	K-tuple = 6 gap = –16, –4	Match = +20, Mismatch = –18	Gene length	358
BLASTN2 (default)	PRI	Word size = 11 gap = –10, –10	Match = +5, Mismatch = –4	Gene length	42
BLASTX (default)	SWISS TREMBL	Word size = 3	BLOSUM62	Tyrosine phosphatase	575 (SWISS) 100 (TREMBL)
FASTX (default)	SWISS TREMBL	K-tuple = 2 gap = –15, –3, frameshift = –30	BLOSUM50	Tyrosine phosphatase	423 (SWISS) 40 (TREMBL)

PRI, Primate division of EMBL (Version 46); SWISS, SWISSPROT protein database (Release 33); TREMBL, translated EMBL database (Version 1); gap, gap insertion penalty, gap extension penalty.

Algorithms used: Smith–Waterman (Smith and Waterman, 1981), implemented on a Biocceleratar at the SEQNET facility, Daresbury; BLAST (Altschul *et al.*, 1990); BLAST2 (BLAST Version 2; Altschul *et al.*, 1997); FASTA (Pearson and Lipman, 1988).

Testing the database searching algorithms

Each of the 1000 artificially evolved sequences was used as a query for database searches. Table 2 gives a summary of the database searching algorithms, parameters and test sets used for each search. Smith–Waterman aligns the query sequence against each sequence in the database, whereas BLAST and FASTA look for exact matches of a specific size (specified by the word size or K-tuple parameters) between the query and database sequences. For each database search, the top hit and corresponding search statistics were recorded. The hit was labelled 'correct' if the evolved sequence matched against the 'seed' sequence or homologue, otherwise it was labelled 'incorrect'. All but one of the search sets were carried out using the default parameters.

Data analysis

Coverage of the database searches

We define the coverage of a database search to be the ability of the algorithm to pick out the appropriate homologous sequence from the database to the query sequence, irrespective of the score given. The coverage of the database searches was defined as the percentage of correct top hits out of all the searches carried out. The effectiveness of database search methods can also be illustrated by the distance the query sequences can be evolved before the search method no longer finds the homologous sequence in the database. We can therefore define a PAM sensitivity for a search algorithm as being the distance, d, to which a seed sequence, $S(0)$, can be evolved before $S(0)$ is no longer the top sequence found using $S(d)$ as a probe in 50% of cases. The larger the PAM sensitivity of an algorithm, the more effective the search method.

Discriminatory power of the database search statistics

Coverage does not take account of whether the database search statistics would be able to recognize the correct match as significant or not. It is perfectly possible for a correct top hit to be given a score which would lead to the hit being ignored. The relative operating characteristic (ROC) curves allowed us to test the usefulness of the search statistics [see Swets and Pickett (1982) and Shah and Hunter (1997)]. The curves were obtained by plotting the sensitivity (P^+) of the searches against the specificity (P^-).

$$\textit{Sensivity}: P^+ = \frac{t^+}{t^+ + f^-} \quad \textit{Selectivity}: P^- = \frac{t^-}{t^- + f^+}$$

where t^+ is a true positive: correct hit, with a score above threshold; f^- is a false negative: correct hit, with a score below threshold; t^- is a true negative: incorrect hit, with a score below threshold; f^+ is a false positive: incorrect hit, with a score above threshold.

For each search carried out using both the artificially evolved and the real sequence, we knew which of these results were correct or incorrect, and the search score given for that hit. If the score for a database hit was above the threshold, the hit was assigned as positive, below the threshold they were assigned as negative. If a specific threshold value was selected, it was therefore possible to assign all hits as either true positives, true negatives, false positives or false negatives. For BLAST probabilities and FASTA expectation values, the converse is true. P^+ and P^- were calculated for a range of threshold values. The range was selected to ensure that P^- ranged from 0 to 1.

The area under the curve was found by numerical integration. This gave us a performance measure P. P is essentially a measure of how well the statistic is able to discriminate between correct and incorrect hits. If $P = 1$, there is a threshold that allows us to discriminate between correct and incorrect hits perfectly. When $0 < P < 1$, there are always going to be false results, whatever threshold is set.

We can define the optimal cut-off to be the one that minimizes the total number of errors. The number of true positives which have been missed is given by $(1 - P^+)$. Similarly, the number of true negatives that have been missed is given by $(1 - P^-)$. Therefore, the optimal cut-off is the one that minimizes $(1 - P^+) + (1 - P^-)$. The optimal cut-off was calculated for each of the database search methods.

Results and discussion

Coverage of the database searches: BLASTN and Smith–Waterman

The results of the database searches using the evolved sequences were plotted. Figure 1 shows the number of correct top-ranked hits out of 10 obtained for each set of sequences of different gene type using the default Smith–Waterman and BLASTN searches. A reverse sigmoidal shape is observed. The same shape curve was obtained for each gene type used, indicating that the search results are independent of gene type. The PAM distance at which only 50% of the correct top hits are found gives the PAM sensitivity, a useful measure of the coverage of a database search method. For BLASTN, the coverage falls to 50% at around PAM 75; for Smith–Waterman, the coverage falls to 50% at around PAM 125. This indicates that the Smith–Waterman default searches perform better than the default BLASTN searches in identifying the more distantly related sequences.

If the per cent identity between the original and evolved DNA sequences was >70%, the hits were significant. However, this was only the case for sequences that had been evolved to PAM 25. For sequences had been evolved to PAM

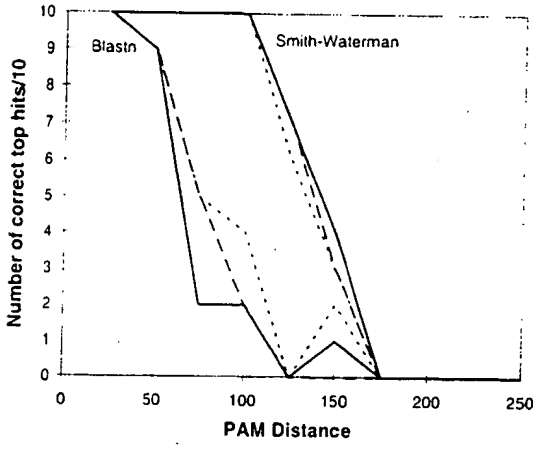


Fig. 1. The number of correct top-ranked hits out of 10 obtained at each PAM distance for each set of sequences of different gene type using the default Smith–Waterman and BLASTN searches. —, ciliary neurotrophic factor; - - -, tyrosine phosphatase; · · ·, retinol binding protein.

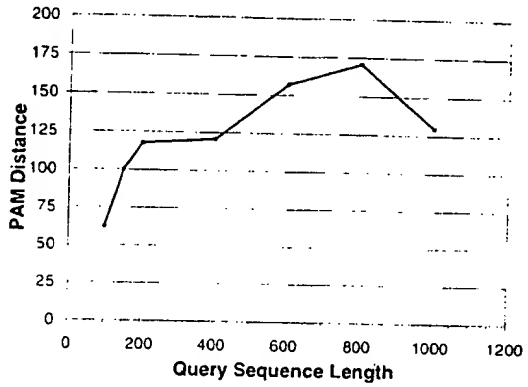


Fig. 2. Effect of query sequence length on the effectiveness of Smith–Waterman searches. The plot shows the average PAM distance of the query sequences when the coverage of the database search is 50% (the PAM sensitivity) as a function of sequence length.

50 or greater, the average per cent identity fell to ~60%, at which point it remained constant as a function of increasing PAM distance (data not shown). This suggests that the percentage identity between the probe sequence and a match in the database is not a good statistic for determining whether a match is significant. This is different from the case of proteins where it is well established that sequence matches showing >25% sequence identity are significant (provided that the probe sequence is long enough). It does not seem to be possible to detect significant matches between DNA sequences when the distance between them is greater than PAM 125. This again is in contrast to protein sequence database searches where the 'twilight zone' is generally considered to occur at about PAM 250, equivalent to 25% sequence identity (see below for a more detailed analysis).

Figure 2 shows the PAM distance at which the coverage falls to 50% as a function of probe sequence length using Smith–Waterman. From this figure, it can be seen that there is a strong effect of probe sequence length of the coverage, with a sharp fall off in performance for probe sequences shorter than 200 bp long. Therefore, we would not expect database searches using a query sequence of <200 nucleotides to be very effective. These results are reassuring in the context of expressed sequence tag (EST) analysis, which plays such an important role in current genome research. ESTs are generally 300–400 nucleotides in length (Boguski *et al.*, 1993), longer than the minimum length suggested here.

Table 3 shows the percentage coverage of the database searches, using the sets of sequences of different length longer than 200 nucleotides. These values provide a measure of the sensitivity of the algorithms. Smith–Waterman searches have the best coverage, and are therefore the most sensitive. BLASTN with a word size of six has the second highest percentage coverage. FASTA had a very similar coverage to that seen for BLASTN with a word size of 6. The worst performing algorithm was the default BLASTN.

Table 3. Percentage coverage (i.e. percentage of correct evolutionary relationships detected) of the different database searching methods using the artificially evolved query sequences of length 200 nucleotides or longer

Database search method	% coverage
BLASTN default	21.6
BLASTN $w = 6$	48.4
BLASTN2 default	23.6
FASTA default	43.4
SW default	58.2

A number of searches using real tyrosine phosphatase nucleotide sequences were carried out against databases containing just one member of the tyrosine phosphatase gene family. The performance measure, P, for the Smith–Waterman and BLAST algorithms (word length = 6) was 0.858001 and 0.876623, respectively. For Smith–Waterman, the discriminatory power of the Z-score statistic was the same as observed for the model data. For BLAST searches, the statistics did not perform quite as well for the real data as they do for the model data.

Shah and Hunter (1997) calculated ROC curves for FASTA Z-score and BLAST expectation to evaluate the performance of these statistics in classifying enzymes according to their International Enzyme Commission (EC) classifica-

tion. They obtained P values of 1 for ~40% of EC classes, and $0.8 < P < 0.99$ for ~48% of classes. These P values were the result of protein database searches and so we would not expect the DNA database search statistics to perform as well because of the degeneracy and smaller alphabet of DNA sequences. However, the P values of 0.85–0.87 for the real DNA data searches are within the range observed for protein database searches. This again suggests that comparing a DNA sequence against a DNA database can produce useful results.

Coverage of the database searches: BLASTX and FASTX

The BLASTX and FASTX searches of the evolved sequences against SWISSPROT gave 50% coverage at PAM 50 (data not shown). This might appear to be an anomolousy low value. However, SWISSPROT did not contain a translation of the original tyrosine phosphatase sequence, only related sequences. For the BLASTX and FASTX searches against TREMBL, 50% coverage was yielded at PAM 210 (data not shown). TREMBL did contain a translation of the original 'seed' sequence. The searches against TREMBL were more sensitive than those against EMBL. This is expected as proteins are non-degenerate and comprise a larger alphabet than DNA. These results demonstrate the importance of searching SWISSPROT, EMBL and TREMBL. TREMBL contains the translations of all coding sequences (CDS) specified in EMBL, but not already in SWISSPROT (Bairoch and Apweiler, 1996). As translations only appear in TREMBL if they have been specified in the nucleotide sequence annotation, it is important to search a nucleotide database such as EMBL, which in any case is the most up to date. These results again demonstrate the fact that searching a DNA sequence against a DNA sequence database can sometimes give better results than using the more traditional FASTX or BLASTX.

Discriminatory power of the database search statistics

Figure 3 shows ROC curves for the BLAST, FASTA and Smith–Waterman searches using the sets of sequences of different lengths (200, 400, 600, 800 and 1000). The values of P^+ and P^-, which were determined over a range of thresholds, are not shown.

The performance measures P for each search method are shown in Figure 3. The BLAST and FASTA searches all give a similar P value (P = 0.96–0.97). This indicates that the search statistics are performing well. The Smith–Waterman searches produce a P value of 0.85. Therefore, although Smith–Waterman is the algorithm which gives the best coverage (see above), it has the least discriminating score for

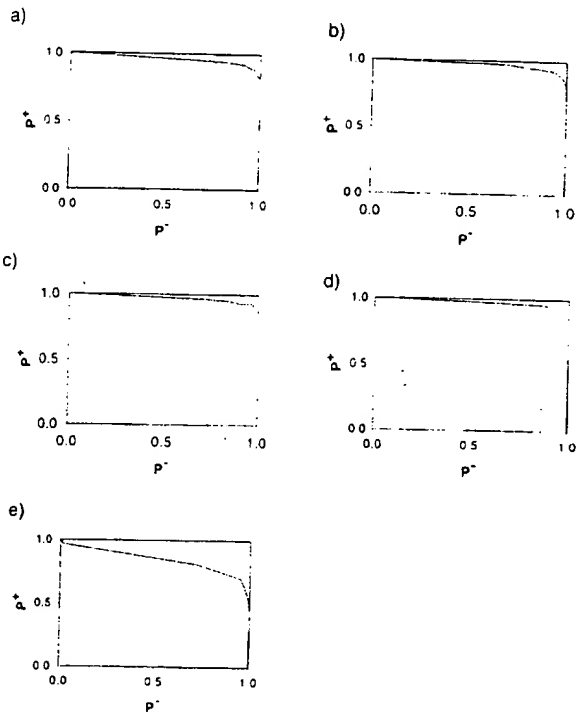


Fig. 3. ROC curves for the DNA database searches using the sets of sequences of different lengths. (**a**) BLASTN searches, default settings. $P = 0.960843$. (**b**) BLASTN searches, using a word size of 6. $P = 0.972293$. (**c**) BLASTN2, default settings. $P = 0.970676$. (**d**) FASTA, default settings. $P = 0.973303$. (**e**) Smith–Waterman, default settings. $P = 0.845113$.

determining whether a top hit is significant. The Z-score does not work as well as the P and E values used by BLAST and FASTA, respectively.

The optimal cut-offs for the database search statistics were calculated by finding a cut-off value for the database search statistic which minimized $(1 - P^+) + (1 - P^-)$. A good default cut-off value to use for the P or E score is either 0.01 or 0.005. At these values of the cut-offs, the rate of false positives is <2% and the rate of false negatives is <5%. For Smith–Waterman searches, the cut-off value for the Z-score should be set at 5. At this value, the rate of false positives is 1.5% and the rate of false negatives is 20%. From these numbers, it is clear that the Smith–Waterman Z-score seriously underestimates positive hits. Altschul *et al.* (1994) provide a useful introduction to the strengths and weaknesses of the different scoring statistics used, and particularly into the problems associated with using the Z-score as a statistic.

Conclusions

Our aims in this paper were to assess the strengths and weaknesses of different DNA database searching algorithms, and to provide guidelines to help assess the significance of the results obtained. Because of the problems of creating suitable test sets, we have had to concentrate on DNA sequences that contain protein coding regions. However, the greatest increase in DNA sequences is currently coming from ESTs and the results from this work should be applicable for this important class of sequences.

The Smith–Waterman algorithm is the best algorithm for finding remote homologues of DNA sequences; however, it suffers from having the least discriminating statistics (*Z*-score). BLASTN run at a word length of 11 (default) has the worst coverage of the algorithms tested; however, it does have the most discriminating statistics. The results show that a sensible compromise for coverage, sensitivity and speed is to use either FASTA or BLASTN with a word length of 6. This gives almost as good a coverage as can be obtained with Smith–Waterman, but with the advantage of better sensitivity. The cut-off for the BLASTN *P* value should be set at 0.01, the cut-off for the FASTA *E* value should be 0.005. Any matches found with scores below these *E* or *P* values should be significant 98% of the time.

Analysis of the results of the test set has suggested optimal cut-off values of the search statistics. These are given below:

BLASTN (default)	*P* value:≤ 0.01
BLASTN ($w = 6$)	*P* value:≤ 0.01
BLASTN2 (default)	*P* value:≤ 0.005
FASTA	*E* value:≤ 0.005
SW	*Z*-score: ≥ 5 (Implementation: Biocellerator bic_sw)

These thresholds should only be applied when a query sequence of at least 200 nucleotides has been used as searches using query sequences shorter than this have been shown to be ineffective. It is important to realize that these scores have been calculated for searches of a database of 70 000 sequences. If the simulations had been run on a larger database, the cut-off values obtained would have been larger. Therefore, these cut-offs are conservative.

A PAM distance of 130 between protein sequences seems to be the limit of sequence homology detection between the corresponding DNA sequences. A distance of PAM 130 corresponds to ~35% amino acid identity between two protein sequences (Dayhoff *et al.*, 1978)—good enough to identify members of the same superfamily. Therefore, comparison of a DNA sequence for a coding region against a DNA database can provide useful structural and functional information on the associated protein product. However, translating the DNA coding region to protein and comparing at the protein level is still the most sensitive way to search for matches—picking up matches at a distance of PAM 210, equivalent to a 20–25% sequence identity at the protein level. However, this only works if either the protein translation of the matching DNA sequence can either be created sensibly on the fly (for algorithms such as BLASTX) or has already been correctly entered into the protein sequence database. This need not necessarily be the case, particularly for EST sequences, in which case a comparison of a DNA sequence against a DNA database can provide useful information.

Acknowledgements

This work benefited from the use of the SEQNET facility. The authors are grateful to Crispin Miller, Paul Higgs and Nick Lawson for useful comments, and to Guy Slater for making his Evolve program available. One of the authors (I.A.) is the grateful recipient of a BBSRC studentship.

References

Altschul,S.F., Gish,W., Miller,M., Myers,E.W. and Lipman,D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Altschul,S.F., Boguski,M.S., Gish,W. and Wooton,J.C. (1994) Issues in searching molecular sequence databases. *Nature Genet.*, **6**, 119–129.

Altschul,S.F., Madden,T.L., Schaffer,A.A., Zhang,J., Zhang,A., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Bairoch,A. and Apweiler,R. (1996) The SWISS-PROT protein sequence data bank and its new supplement TREMBL. *Nucleic Acids Res.*, **24**, 21–25.

Benner,S.A., Cohen,M.A. and Gonnet,G.H. (1993) Empirical and structural models for insertions and deletions in the divergent evolution of proteins. *J. Mol. Biol.*, **229**, 1065–1082.

Boguski,M.S., Lowe,T.M.J. and Tolstoshev,C.M. (1993) dbEST—database for 'expressed sequence tags'. *Nature Genet.*, **4**, 332–333.

Bork,P. and Bairoch,A. (1996) Go hunting in sequence databases but watch out for the traps. *Trends Genet.*, **12**, 425–427.

Dayhoff,M.O., Schwartz,R.M. and Orcutt,B.C. (1978) A model of evolutionary change. In Dayhoff,M.O. (ed.), *Atlas of Protein Sequence and Structure*. National Biomedical Research Foundation, Washington, DC, Volume 5, Suppl. 3, pp. 345–352.

Mann,M. (1996) A shortcut to interesting genes: peptide sequence tags, expressed-sequence tags and computers. *Trends Biochem. Sci.*, **21**, 494–495.

Murzin,A.G., Brenner,S.E., Hubbard,T. and Chothia,C. (1995) SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, **247**, 536–540.

Pearson,W.R. and Lipman,D.J. (1988) Improved tools for biological sequence comparison. *Proc. Natl Acad. Sci. USA*, **85**, 2444–2448.

Sander,C. and Schneider,R. (1991) Database of homology derived protein structures and the structural meaning of sequence alignment. *Proteins*, **9**, 56–68.

Shah,I. and Hunter,L. (1997) Predicting enzyme function from sequence: a systematic appraisal. In Gaasterland,T., Karp,P., Karplus,K., Ouzounis,C., Sander,C. and Valencia,A. (eds), *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, Menlo Park, CA, pp. 276–283.

Shpaer,E.G., Robinson,M., Yee,D., Candlin,J.D., Mines,R. and Hunkapiller,T. (1996) Sensitivity and selectivity in protein similarity searches: a comparison of Smith-Waterman in hardware to BLAST and FASTA. *Genomics*, **38**, 179–191.

Slater,G. (1995) Modelling of molecular evolution as an approach to a quantitative evaluation of sequence comparison algorithms. MSc Thesis, University of Manchester.

Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.

Swets,J. and Pickett,R.M. (1982) *Measuring the Accuracy of Diagnostic Systems*. Academic Press, New York.

BMC Bioinformatics

BioMed Central

Methodology article

Open Access

Fast and sensitive multiple alignment of large genomic sequences

Michael Brudno*[1], Michael Chapman[2], Berthold Göttgens[2], Serafim Batzoglou[1] and Burkhard Morgenstern*[3,4]

Address: [1]Department of Computer Science, Stanford University, Stanford, CA 94305, USA, [2]Department of Haematology, University of Cambridge, Cambridge Institute for Medical Research, Hills Road, Cambridge CB2 2XY, United Kingdom, [3]International Graduate School in Bioinformatics and Genome Research, Universität Bielefeld, Postfach 100131, 33501 Bielefeld, Germany and [4]University of Göttingen, Institute of Microbiology and Genetics, Goldschmidtstr. 1, 37077 Göttingen, Germany

Email: Michael Brudno* - brudno@cs.stanford.edu; Michael Chapman - mac54@cus.cam.ac.uk; Berthold Göttgens - bg200@cam.ac.uk; Serafim Batzoglou - serafim@cs.stanford.edu; Burkhard Morgenstern* - bmorgen@gwdg.de

* Corresponding authors

Published: 23 December 2003

BMC Bioinformatics 2003, 4:66

Received: 01 September 2003

Accepted: 23 December 2003

This article is available from: http://www.biomedcentral.com/1471-2105/4/66

Abstract

Background: Genomic sequence alignment is a powerful method for genome analysis and annotation, as alignments are routinely used to identify functional sites such as genes or regulatory elements. With a growing number of partially or completely sequenced genomes, *multiple alignment* is playing an increasingly important role in these studies. In recent years, various tools for pair-wise and multiple genomic alignment have been proposed. Some of them are extremely fast, but often efficiency is achieved at the expense of sensitivity. One way of combining speed and sensitivity is to use an *anchored-alignment* approach. In a first step, a fast search program identifies a chain of strong local sequence similarities. In a second step, regions between these anchor points are aligned using a slower but more accurate method.

Results: Herein, we present CHAOS, a novel algorithm for rapid identification of chains of local pair-wise sequence similarities. Local alignments calculated by CHAOS are used as anchor points to improve the running time of DIALIGN, a slow but sensitive multiple-alignment tool. We show that this way, the running time of DIALIGN can be reduced by more than 95% for BAC-sized and longer sequences, without affecting the quality of the resulting alignments. We apply our approach to a set of five genomic sequences around the stem-cell-leukemia (SCL) gene and demonstrate that exons and small regulatory elements can be identified by our multiple-alignment procedure.

Conclusion: We conclude that the novel CHAOS local alignment tool is an effective way to significantly speed up global alignment tools such as DIALIGN without reducing the alignment quality. We likewise demonstrate that the DIALIGN/CHAOS combination is able to accurately align short regulatory sequences in distant orthologues.

Background

Cross-species sequence comparison is playing an increasingly important role in genome analysis and annotation, see [1-3] for review. The functional parts of genomes are under selective pressure, and therefore evolve more slowly than non-functional parts, where random mutations can be tolerated without affecting the evolutionary fitness of the organism. Consequently, conserved sequences often correspond to functional elements. Comparative sequence analysis has been used for a variety of purposes,

e.g. gene prediction [4-10], identification of regulatory elements [11-17] and identification of signature sequences to detect pathogene microorganisms [18]. One major advantage of comparative approaches is that they are based on simple measurement of sequence similarity and require little additional information about the features to be detected. While more traditional methods need large sets of training data to construct species-specific statistical models of genes or regulatory elements, comparative methods essentially depend on the availability of syntenic sequences at an appropriate evolutionary distance, making them effective for analysis of newly sequenced genomes, when little training data is available.

In recent years, a number of algorithms have been proposed for pair-wise genomic alignment; these algorithms combine local and global alignment features by returning ordered chains of local similarities. Some approaches use suffix-tree or hashing algorithms to identify pairs of *k*-mers of a certain minimum length (and, possibly, a maximum number of mismatches) [19-21]. These methods are extremely time-efficient but are most effective at aligning sequences from closely related genomes, e.g. from different strains of a bacterium [19]. A more flexible approach has been implemented in the PipMaker [22] set of tools, where a local alignment program implementing a gapped BLAST algorithm, BLASTZ [23], is used.

A sensitive and versatile tool for *multiple* alignment of distal sequences is DIALIGN [24]. Originally, this approach has been developed to align protein and DNA sequences of limited length, e.g. [25], but in more recent studies the program has also been applied to large genomic sequences. Göttgens *et al.* [14,15] used DIALIGN to detect small regulatory sites in vertebrate genome sequences. Fitch *et al.* identified consensus sequences in pathogen viral genomes based on DIALIGN multiple alignments; these consensus sequences were used to identify sequence *signatures* for pathogen detection [18]. Unfortunately, the use of DIALIGN for analysis of genomic sequences has been limited by the long program running time: the original algorithm for pair-wise alignment required time proportional to the product of the lengths of the input sequences [26], which is too slow for long sequences.

One way of combining speed and sensitivity for genomic alignment is to use an *anchored-alignment* approach. In a first step, a fast search tool is used to identify a chain of high-scoring sequence similarities. These similarities are then used as anchor points for the final alignment, where a more sensitive method aligns those regions that are left over between the identified anchor points. Such an approach was initially proposed by Batzoglou *et al.* [6]. These authors developed *GLASS*, a system that aligns genomic sequences based on matching *k*-mers. Obviously, the more dense a chain of anchor points is, the higher is the reduction of the search space and gain in speed for the final procedure – on the other hand, too many anchor points could overly restrict the search space, leading to decreased alignment quality. The main challenge in the anchored-alignment approach is therefore to find a trade-off between speed and alignment quality – to locate anchor points that are as dense as possible while still leading to optimal or near-optimal alignments.

Results

In this section we first describe the CHAOS procedure for local alignment of two sequences. We then explain how pair-wise similarities identified by CHAOS can be used as anchor points for pairwise or multiple alignment. Finally, we evaluate our approach in detail, using pair-wise and multiple test data sets.

CHAOS local alignment algorithm

The CHAOS algorithm works by chaining together pairs of similar regions, one from each of the two input DNA sequences; we call such pairs of regions *seeds*. More precisely, a seed is a pair of words of length k with at least n identical base pairs (*bp*). A seed $s^{(1)}$ can be chained to another seed $s^{(2)}$ whenever (i) the indices of $s^{(1)}$ in both sequences are higher than the indices of $s^{(2)}$, and (ii) $s^{(1)}$ and $s^{(2)}$ are "near" each other, with "near" defined by both a distance and a gap criteria as illustrated in Figure 1. The final score of a chain is the total number of matching *bp* in it. The default parameters used by CHAOS are words of length 10, with a degeneracy of one (n = k-1), a distance and gap criteria of 20 and 5 bp respectively, and a score cutoff of 25. The detailed algorithms used for finding seeds and computing the maximal chains are specified in Methods.

After computing the maximal chains, CHAOS scores each chain by using match and mismatch penalties for the letters of each seed. For two seeds seperated by x and base y pairs in the first and second sequences, a gap penalty proportional to $|x - y|$ is incurred. CHAOS throws away chains that score below some threshold t. We augment this scoring method, by adding a rapid rescoring step: chains that score below t are immediately thrown away. Chains that score above t are rescored by performing ungapped extensions in both directions from each seed, and finding the optimal location to insert exactly one gap of size $|x - y|$. The matches and mismatches can be scored with an arbitrary substitution matrix. CHAOS can be used as a standalone program for local sequence alignment or as a preprocessing step to find anchor points for global alignment procedures.

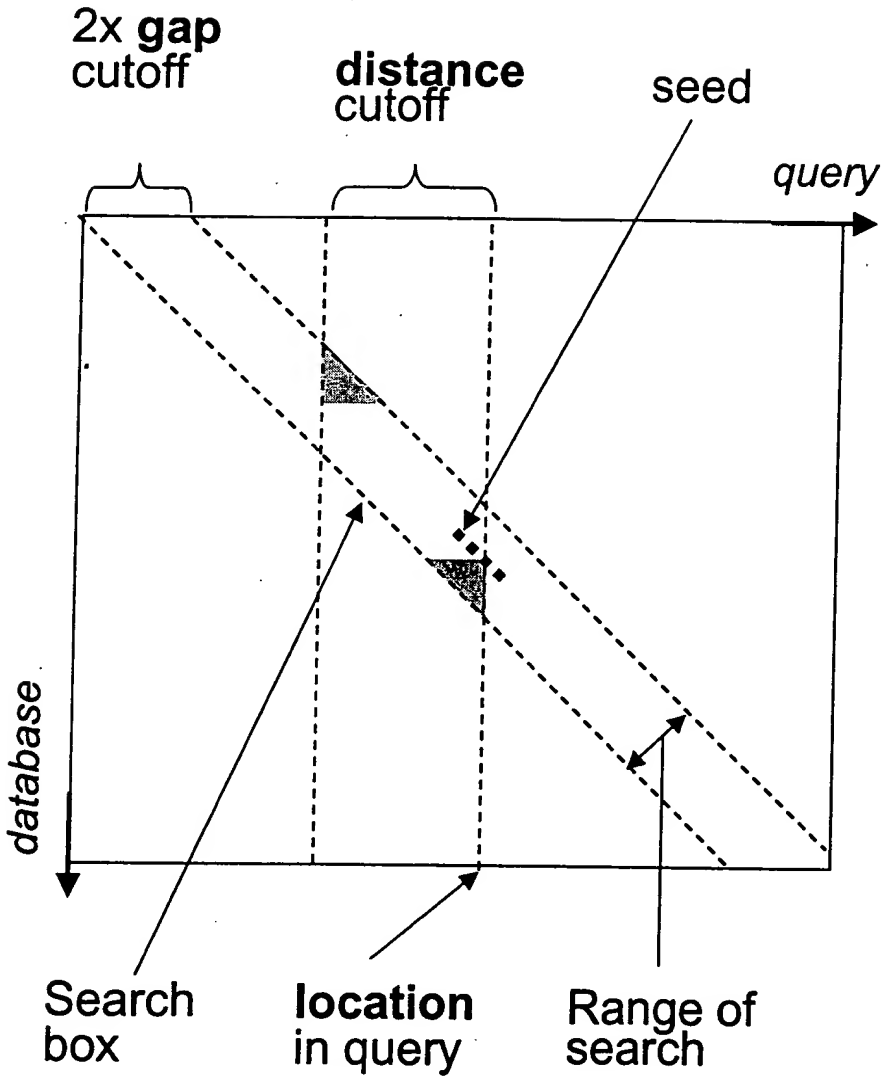


Figure 1
The figure shows a matrix representation of sequence alignment. The seed shown can be chained to any seed which lies inside the *search box*. All seeds located less then *distance bp* from the *current location* are stored in a skip list, in which we do a range query for seeds located within a *gap* cutoff from the diagonal on which the current seed is located. The seeds located in the grey areas are not available for chaining to make the algorithm independent of sequence order.

Anchored pair-wise and multiple alignment

In the present study, we use CHAOS to identify *chains* of local sequence similarities that can be used as anchor points for DIALIGN. Once CHAOS has identified a collection of local alignments for a pair of input sequences, we use an algorithm based on the longest increasing subsequence problem [27] to find the highest scoring chain of local alignments in time $O(N \log N)$, where N is the number of local alignments. For *pair-wise* alignment, this chain is directly used to anchor the DIALIGN alignment as described in [28].

For anchored *multiple alignment*, we proceed as follows: in a first step, we apply CHAOS to all possible pairs of input sequences; this way we obtain a list of similarities that we consider as *candidate* anchor points. The problem with these similarities is that they may contradict each other, *i.e.* it may not be possible to include all of them simultaneously in one single multiple alignment. To solve this *consistency* problem, we use the same greedy algorithm that DIALIGN uses to find consistent sets of local pairwise alignments in the process of multile alignment calculation [29]. A quality score is associated with each of the identified candidate anchors and the set of all candidate anchors is sorted by these scores. Starting with the highest-scoring one, candidate anchors are accepted one-by-one as final anchor points – provided they are consistent with those candidates that have been accepted previously. Non-consistent similarities are discarded. This way, we finally obtain a consistent set of pair-wise anchor points, *i.e.* a set of anchor points that would fit into one single multiple alignment, see also [29,24,30] where our greedy procedure is explained in the context of the DIALIGN algorithm.

Program Evaluation

It is common practice to evaluate sequence alignment programs by applying them to real-world sequences with known functional sites or 3D structure. For protein alignment, several sets of benchmark sequences are available [31-33]; they are routinely used as standards of truth to evaluate and compare the performance of multiple alignment programs. For pair-wise comparison of genomic sequences, benchmark data have been compiled by Jareborg *et al.* [12] and Batzoglou *et al.* [6], these data have been used for comparative gene finding. So far, however, there are no generally accepted reference data with which to evaluate software programs for *multiple* genomic alignment. Herein, we first use the Jareborg benchmark data to demonstrate that our anchored-alignment procedure improves the running time of DIALIGN by up to two orders of magnitude while the resulting alignments are essentially the same as with the original non-anchored algorithm. Secondly, we apply our method to a set of five genomic sequences around the stem-cell-leukemia (SCL) gene. For all evaluations we start by masking the repeats in the sequences with RepeatMasker. We analyze the resulting multiple alignment in detail and we show that not only is the speed of DIALIGN is improved, but also important functional elements missed by the original DIALIGN can be detected by using the CHAOS anchors. Additional multiple sequence sets are used to demonstrate how the improvement in running time that we achieve depends on the length of the input sequences.

Running time for pair-wise alignment

The Jareborg data set consists of 42 annotated sequence pairs from human and mouse varying in length between less than 6 *kb* and more than 227 *kb*, with an average length of 38 *kb*. These sequences have been used in a paper for a systematic comparison of five different genomic alignment programs [10]. The result of this previous study was that DIALIGN was superior to other methods in terms of alignment quality, but inferior in terms of running time. Since these results have been published previously, we do not repeat the evaluation of DIALIGN for pair-wise alignment. Instead, we focus on how our anchoring procedure affects running time and alignment quality compared with the non-anchored DIALIGN.

We first applied CHAOS to our data in order to obtain chains of anchor points. Next, we aligned the sequence pairs with DIALIGN, first without anchoring and then using the anchor points identified by CHAOS, and we compared the program running time and quality of the resulting alignments. DIALIGN was run with the *translation* option where local similarity among DNA sequences is compared at the peptide level, see [29]. When CHAOS is run with default parameters the density of the returned anchor points was, on average, 2.1 anchor points per *kb*. The results in terms of alignment quality and program running time are summarized in Table 1. With a *cutoff* value of 20 for CHAOS, the program running time of the anchored DIALIGN could be improved by 95% compared to the non-anchored program, while the scores of the resulting alignments were reduced by about 1%. Alignment quality was measured at two distinct levels, (*a*) by considering the *numerical* score of the produced alignments and (*b*) by considering their *biological* quality. To this end, alignments were compared to annotated protein-coding exons and sensitivity and specificity were measured at the nucleotide level, i.e. a nucleotide that is part of a selected fragment is considered a *true positive* (TP) if it is also part an annotated exon and as *false positives* (FP) if it is not; true and false negatives (TN and FN) are defined accordingly. We used the usual measures for prediction accuracy, namely *sensitivity* = TP/(TP + FN), specificity = *TP/(TP + FP)*, and *approximate correlation* = 0.5 ((TP/(TP + FN)+(TP/(TP + FP)+(TN/(TN + FP)+(TN/(TN + FN))-1.

Table 1: Total CPU time and alignment quality for DIALIGN (D) and DIALIGN anchored with CHAOS (C+D) applied to a set of 42 pairs of genomic sequences from human and mouse [12]. CHAOS was run with varying cutoff parameters. Lower cutoff values for CHAOS produced higher numbers of anchor points resulting in a decreased search space for the final DIALIGN alignment procedure thus leading to improved running time but slightly decreased alignment quality. The average number of anchor points per kilobase is shown (anc./kb). *Score* is the total *numerical* score of all produced DIALIGN alignments, *i.e.* the sum of the scores of the segment pairs in the alignments. As a rough measure of the *biological* quality of the produced alignments, we compared local sequence similarities identified by DIALIGN and CHAOS to known protein-coding regions. Here, Sn, Sp and AC are *sensitivity*, *specificity* and *approximate correlation*, respectively. For the D and C+D results, DIALIGN was evaluated by comparing *all* segment pairs contained in the alignment to annotated exons.

program	cutoff	anc./kb	CPU	%CPU	score	%score	Sn	Sp	AC
D			179,001	100.0	54,214	100.0	83	40	57
C+D	35	1.4	14,334	8.0	53,839	99.3	83	40	57
C+D	30	1.7	11,717	6.5	53,820	99.2	83	40	57
C+D	25	2.1	11,485	6.4	53,654	98.9	83	40	57
C+D	20	2.8	8,964	5.0	53,642	98.9	83	40	57
C+D	15	4.2	7,404	4.1	53,208	98.1	82	41	57
C+D	10	6.5	6,696	3.7	52,684	97.1	82	41	57

Multiple alignment of the stem-cell-leukemia (SCL) region

To test the combined CHAOS-DIALIGN algorithm for multiple alignment, we used a set of five genomic sequences around the stem cell leukaemia (SCL) gene. SCL is a critical regulator of haematopoiesis, with a pattern of expression that is conserved in all species studied, from mammals to teleost fish [34]. Locations of the exons and of a number of important regulatory regions have been previously experimentally determined. We took SCL sequence from immediately after the upstream gene to the end of the sequence or just after the downstream gene – whichever was longer – in five species: human, mouse, chicken, pufferfish, and zebrafish. We aligned these with DIALIGN, both with and without prior CHAOS anchoring. We then examined the alignments for regions of sequence conservation between all five species.

A total of 265,145 bases were aligned. With a new *mixed-alignment* option and the -o option, the combined CHAOS-DIALIGN algorithm completed the task in 1 hour and 35 minutes while the non-anchored DIALIGN took 6 hours and 6 minutes. *Mixed-alignments* means that local similarities are evaluated in two ways, at the *nucleotide* level and at the *peptide* level where segments are translated according to the genetic code and the resulting peptide segments are compared. This option is appropriate where genomic sequences are aligned that may contain coding as well as non-coding homologies but it is relatively time consuming. The -o option is used for reduced running time, see the DIALIGN user guide for details. By contrast, if our sequences were compared at the peptide level only, the running time was 13.8 minutes with the anchoring procedure and 49.2 minutes with the non-anchored version of DIALIGN. These test runs were carried out on a Linux PC with a 2.4 GHz Pentium 4 processor. With both program options, the running-time improvement achieved by CHAOS anchoring procedure was more than 70 percent while the *numerical* score of the output alignments differed by less than 1 percent ('translated' option) and less than 0.1 percent ('mixed alignment' option).

Of the four fish SCL exons, all of which have homologues in the higher species [35,15], the three coding exons were successfully aligned across all species by both algorithms. The downstream gene, membrane associated protein-17 (MAP17), is not present in pufferfish and contains four, rather short, exons. Moreover, the chicken sequence only extends to the first of these. It is therefore perhaps not surprising that these were only aligned between human and mouse by both algorithms. Within the non-coding DNA, one further region of homology across all species was identified (see Figure 2). This region just upstream of exon 1 has promoter activity in haematopoietic cell lines and also contains a midbrain enhancer [36-38]. Within this region and in all species, CHAOS-DIALIGN perfectly aligned five motifs, each of which is essential for the appropriate pattern or level of SCL transcription [36-39]. Unanchored DIALIGN misaligned the first GATA binding site; otherwise, alignments of the SCL promoter were identical. In the immediate downstream region, within the non-coding exon 1, a further motif was identified by CHAOS-DIALIGN alone. This represents a perfect binding consensus (5 -AANATGGC-3) for the zinc finger transcription factor YY1 [40]. This motif was conserved in all five species and may act as a transcriptional enhancer for the nearby promoter. Alternatively, it may be an RNA-binding element involved in post-transcriptional processing. There is one further non-coding sequence known to be conserved in the five species, but which is not aligned by either DIALIGN algorithm – the AAUAAA

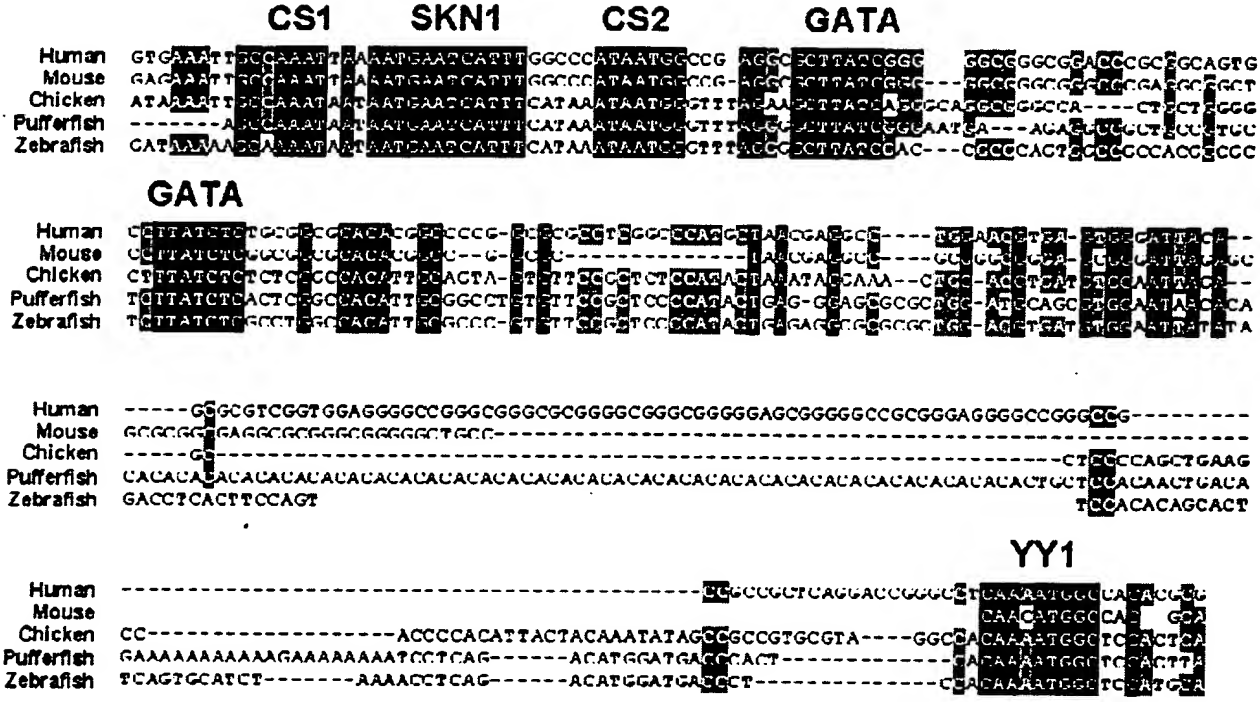


Figure 2
CHAOS-DIALIGN correctly aligns the SCL promoter and a conserved non-coding sequence in exon 1. The alignment was extracted from the CHAOS-DIALIGN global alignment of SCL sequences from human, mouse, chicken, zebrafish, and pufferfish. Consensus binding motifs are labelled. All except YY1 have been previously demonstrated to be essential for the appropriate pattern or level of SCL expression. The factors binding conserved sequence (CS) 1 and 2 are unknown. Shading of bases is at (grey) and (black) conservation.

polyadenylation sequence [15]. However, previous alignment of this region was only possible with *a priori* knowledge of its existence and following extraction and local alignment of the relevant sequences. Other multiple alignment algorithms (MAVID [41], LAGAN [42]) also fail to align this region. It is interesting to note that in two cases the CHAOS/DIALIGN combination produces biologically superior alignments than unanchored DIALIGN. This is likely due to the anchor points limiting the search area of DIALIGN and not allowing it to accept a numerically superior alignment that is incorrect biologically.

Running time for longer sequences

We wanted to explore how the relative improvement in program running time that we achieved by our anchoring method depends on the length of the input sequences. The main benefit of reduced running time of DIALIGN is that this way the program becomes applicable to genomic sequences that were previously beyond its scope, so we wanted to estimate the behavior of the running time for very long sequences. It has been previously shown in [42] that given certain assumptions about the distribution of anchor points on the sequences the running time of an anchored alignment algorithm would be linear in the sequence lengths. In reality, it is difficult to predict the distribution of distances between anchor points since this depends, of course, on the sequences being compared. Nevertheless, for our data we could confirm that the relative improvement in running time for pairwise sequences was far more significant for longer sequences than for shorter ones (Figure 3).

The SCL sequences that we used as a test example for multiple alignment were only 53 *kb* in length, so we did two additional test runs to test the performance of our approach for longer, multiple sequence sets. First, we applied the anchored and non-anchored procedures to a set of three genomic sequences from human, mouse and dog from the interleukin region [43] with an average length of 222 *kb*. We used the *translation* option together with the -o option. Without anchoring, the running time of DIALIGN was 8 *h* and 36 *min* ; with anchor points

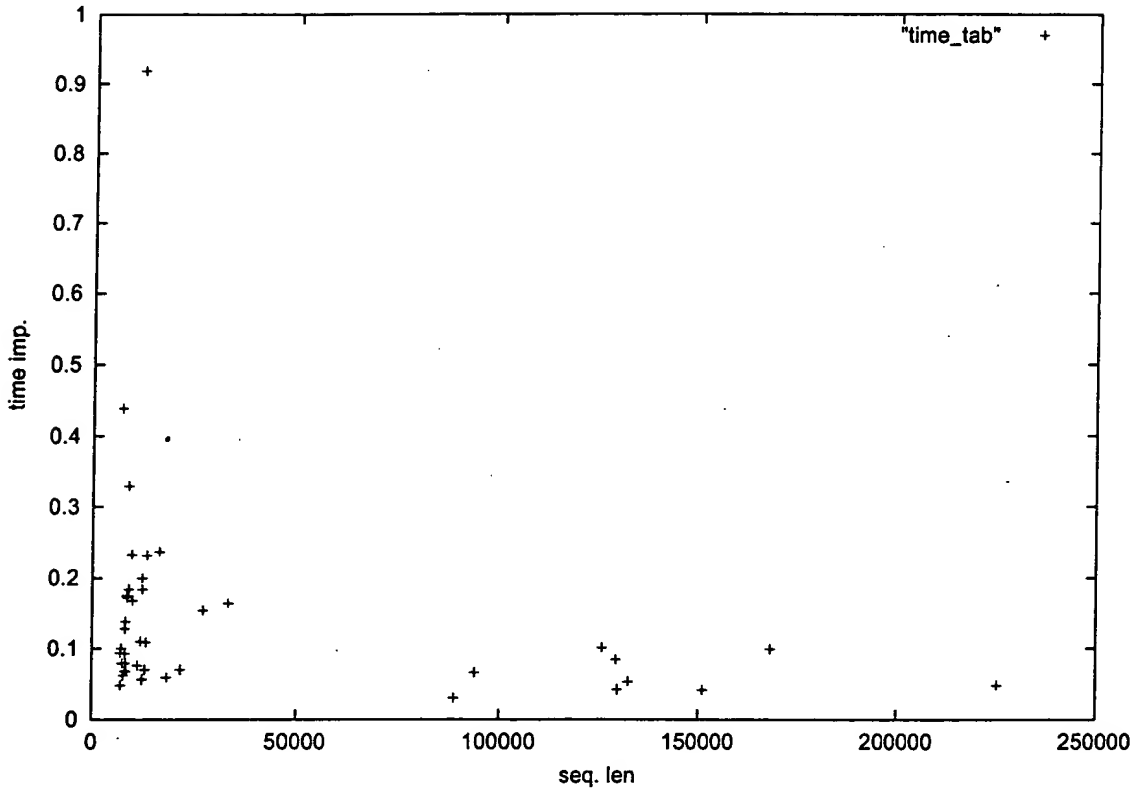


Figure 3
Relative improvement in program running time for 42 pairs of genomic sequences form human and mouse of different length. Each point represents one sequence pair. The *x*-axis is the medium sequence length of sequence pairs while the *y*-axis is the relative running time of the anchored-alignment procedure compared to the non-anchored procedure.

created by CHAOS, the running time was reduced to only 24 *min* and 40 *s*, so the CPU time was reduced by more than 95%. At the same time all the annotated features (all exons and known reguatory sequences) were properly aligned. The numerical score of the anchored alignment was 1.5% below the score of the non-anchored alignment. As a third example, we aligned syntenic sequences from human chromosome 20, mouse chromosome 2 and rat chromosome 3 that had an average length of more than 1 *MB*. The anchored program run terminated after 8 *h* and 17 *min*. We did not complete the non-anchored run but based on the first 2 days we estimated that without anchoring, the program would have terminated after 18 days, so for these sequences, the running time was reduced by around 98%.

Discussion

Multiple alignment of large genomic sequences is now a crucial tool for genome data analysis and annotation. Several studies demonstrated that DIALIGN is a highly efficient and versatile tool for this purpose. It has been used to identify biological relevant signals in raw sequence data, such as regulatory elements [14,16,44,45] or protein-coding regions [10] and a new gene-prediction program called AGenDA (Alignment-based Gene Detection Algorithm) has been developed that relies on DIALIGN alignments as input information [9,46]. Most recently,

DIALIGN has been succesfully used to identify signature patterns for pathogen microorganisms [18]. However, DIALIGN was originally designed to align protein and short DNA sequences and its application to genomic sequences was severely limited by the long program running time. To make the program applicable to larger sequences, we implemented an *anchored-alignment* option where pre-defined anchor points can be used to reduce the search space and running time of the alignment procedure. To identify appropriate anchor points, we developed a fast similarity search tool called *CHAOS*. With the new anchoring option and anchor points created by CHAOS, DIALIGN can now be applied to data sets that were previously beyond its scope.

Most of the methods for heuristic local alignment, such as BLAST [47] and FASTA [48] were developed when the bulk of available sequence were proteins. It has been shown that such algorithms are not as efficient in aligning non-coding sequences [49]. With the new availability of genomic sequences it is appropriate to refine the algorithms used for local alignment so that they more closely reflect the fashion in which the genomic sequences are conserved. Unlike other fast algorithms for genomic alignment, CHAOS does not depend on long exact matches, does not require extensive ungapped homology, and allows mismatches in seeds, all of which are important when comparing distantly related organisms or non-coding regions, where conservation is generally much poorer than in coding areas.

Some previous algorithms for anchored global alignment have worked by first identifying very strong local similarities among the input sequences and adding weaker similarities later. The problem with this approach is that one high-scoring spurious match can lead to a wrong output alignment while many weaker but biologically important homologies may be missed. By contrast, CHAOS searches for the *highest scoring* chain of local alignments. This way, a numerically high-scoring but biologically wrong local alignment can be conterbalanced by a chain of several weaker local alignments - provided that the total score of these alignments exceeds the score of the one wrong alignment.

We demonstrate that the chains of local alignments returned by CHAOS can be used to anchor the DIALIGN alignment procedure, significantly improving the alignment speed, without affecting the quality of the output alignments. To compare the quality of the anchored and non-anchored alignments, we applied both versions of the program to a database of genomic sequence pairs from human and mouse. We compared the *numerical* scores of the resulting alignments as well as their *biological* quality. For *multiple* genomic alignment, no benchmark data are presently available to compare the perfomance of different alignment algorithms systematically. However, the first step in the DIALIGN multiple-alignment procedure is the pair-wise alignment of all possible pairs of input sequences; fragments of these pair-wise alignments are then used to assemble a multiple alignment. Thus, the results that we obtained for pair-wise alignment can be directly applied to multiple alignment.

We could confirm this in a detailed study of a set of five genomic sequences around the *stem cell leukemia (SCL)* gene from vertebrates ranging from fish to human. As with our test runs for pair-wise alignment, the anchoring procedure led to a considerable improvement in running time while the output alignments were virtually the same as without anchoring. The *numerical* scores of the anchored multiple alignments differed by less than 1 percent from the scores of the non-anchored alignments and, again, the *biological* quality of the anchored alignments was even improved. For the SCL sequences, the improvement in running time was less dramatic than with the human-mouse seuqence pairs used to evaluate the pair-wise alignment procedure. There are two obvious reasons for this result. (*a*) The SCL sequences are shorter than the sequences used for pair-wise alignment and, as discussed above, the relative improvement in running time increases with sequence length. (*b*) The SCL sequences are more distantly related than the human-mouse sequence pairs. Thus, the *density* of anchoring points identified by CHAOS is lower than in the previous examples.

In the SCL example, we demonstrated that our method is able to identify small regulatory elements. It should be mentioned that there are a number of limitations associated with distal species comparisons for the identification of putative regulatory regions. In the SCL locus, many known mammalian enhancers cannot be identified in chicken or fish species [15,14]. This may be because sequence divergence is so extensive as to mask short regulatory motifs. In support of this is the observation that some functional regions (e.g. exon 1 and the polyA site) could be aligned only with *a priori* knowledge of their location, extraction of the surrounding sequence, and subsequent local alignment [15]. Alternatively, it may be because regulatory mechanisms differ. An example of this is provided by the enhancer of the IgH locus in catfish, which is capable of activity in mammalian transgenics, but which differs both in its location and critical regulatory motifs between fish and mammals [50]. Where non-coding homology in distal comparisons exists, it is usually a powerful indicator of the presence of a regulatory region. The CHAOS-DIALIGN algorithm was capable of detecting the SCL promoter in a five-way alignment of sequences from human, mouse, chicken, pufferfish, and zebrafish. Furthermore, it correctly aligned all the critical

motifs within this region, and a further YY1 motif in exon 1. As discussed above, homology in all five species for this latter motif has only previously been demonstrated following extraction and local alignment of the relevant sequences using DIALIGN [15]. Other multiple alignment algorithms (MAVID [41], LAGAN [42]) fail to align this motif. Therefore, with the SCL dataset, the quality of the CHAOS-DIALIGN output in terms of biological relevance is superior to that of other multiple global alignment tools. It is also better than that of unanchored DIALIGN and, at the same time, the anchored program is between one and two orders of magnitude faster.

Finally, we want to emphasize the need for further work in the general area of multiple alignments. Perhaps the most pressing problem right now is the inability of researchers to evaluate the alignment programs except by looking at examples which have been annotated by biologists. At the same time the methods that simulate evolution of DNA sequences, such as ROSE [51], are unable to create biologically realistic sequences. Thus it is necessary to create some measure of alignment quality that is based on real sequences without biological annotation.

Conclusion

In this paper, we present a fast local pair-wise alignment tool called *CHAOS* (CHAins Of Seeds); we use this program to speed up the DIALIGN program. For a pair of input sequences, CHAOS returns a chain of local sequence alignments that can be used as anchor points to reduce the search space and running time of any sensitive global alignment procedure: it has also been used for anchoring in the LAGAN [42] alignment tool. We extend the anchoring approach to the problem of *multiple* alignment of large genomic sequences. Multiple alignments are likely to contain much more information about functional sites than pair-wise alignments, and with the increasing amount of genome sequence data, the development of methods for multiple alignment is a high priority.

Systematic test runs with pair-wise alignments demonstrate that this way the running time of DIALIGN can be reduced by one to two orders of magnitude while the quality of the resulting alignments is only minimally affected. Moreover, the relative improvement in speed increases with the length of the input sequences, making our approach particularly effective for alignment of large genomic sequences.

We also applied CHAOS/DIALIGN to a set of five genomic sequences from human, mouse, chicken, zebrafish, and pufferfish around the stem-cell-leukemia (SCL) locus. Our method correctly aligned three coding exons and five motifs involved in transcription regulation. To make our method easily available for the scientific community, we set up an internet server where CHAOS/DIALIGN can be used through a WWW interface.

Methods

In this section we describe the details of the CHAOS local alignment algorithm.

Finding the seeds

Formally, a seed is a pair of words of length k with at least n identical base pairs (*bp*). The seeds are located using a simplified version of the Aho-Corasick [52] algorithm. A variation on the *trie* data structure [53] which we call a *threaded trie* (T-trie) is used to store the k-mers of one sequence. A trie is a tree for storing strings in which there is one node for every common prefix. A node which corresponds to the word $w_1...w_p$ would have as its parent a node that corresponds to $w_1...w_{p-1}$. A trie that contains all of the k-mers of some string has each leaf at depth k, and each leaf stores all of the locations where this k-mer occurs in the indexed sequence.

A T-trie differs from a regular trie in that a node that corresponds to the string $w_1...w_p$ will also have a *back pointer* to the node which corresponds to $w_2...w_p$. We start by inserting into the T-trie all of the k-mers of one of the sequences, which we will call the *database*. Then we do a "walk" using the other, *query* sequence, where we start by making the root of the T-trie our current node, and for every letter of the query:

1. If the *current* node has a child corresponding to this letter we make this child our current node, and return any seeds stored in it,

2. Otherwise make the node pointed to by our *back pointer* our *current* node, and return to step 1.

As an illustration of why this method works well in practice, assume that all of the possible k-mers are present in the database (which is most likely the case). Then, finding the k-mers that correspond to the next letter of the query requires only two pointer operations: the first is to follow a back pointer from the k level node which is our *current* node, the second to follow a down pointer from the resulting node to the appropriate child. Because in practice most k-mers will be present in the database sequence this process will work quickly. To allow degeneracy we permit multiple current nodes, which correspond to the possible degenerate words. It also offers a space saving over the traditional Aho-Corasick automaton as it requires the storage of one rather than four "failure links".

Chaining the seeds

A seed $s^{(1)}$ can be chained to another seed $s^{(2)}$ whenever (i) the indices of $s^{(1)}$ in both sequences are higher than the

indices of $s^{(2)}$, and (ii) $s^{(1)}$ and $s^{(2)}$ are "near" each other, with "near" defined by both a distance and a gap criteria as illustrated in Figure 1.

To find the chains of seeds we use the following algorithm. Let D be the maximum distance between two adjacent seeds. The seeds generated while examining the last D base pairs of the query sequence are stored in a skip list, a probabilistic data structure that allows for fast searches and easy in-order traversal of its elements [54]. The seeds are ordered by the difference of its indices in the two sequences (*diagonal number*). For each seed s found at the *current location* do a search in the skip list for previously stored seeds which have diagonal numbers within the permitted gap criterion of the diagonal number of s. We thus find the possible previous seeds with which s can be chained. The highest scoring chain is picked, and this chain can be further extended by future seeds. In order to enforce the distance criterion we then remove from the skip list all seeds which were generated D base pairs from the positions of the new seeds, and insert the new seeds into the skip list.

Availability and requirements

The combined CHAOS-DIALIGN software is available online at *Göttingen Bioinformatics Compute Server (GoBiCS)*: http://dialign.gobics.de/chaos-dialign-submission

The source code for CHAOS is available at: http://www.cs.stanford.edu/~brudno/chaos/ together with a PERL script that transforms CHAOS output to the format that can be used to anchor DIALIGN. A version of DIALIGN that accepts such anchors is available at: http://bibiserv.techfak.uni-bielefeld.de/dialign/

Authors Contribution

MB developed CHAOS and drafted parts of the manuscript. MC and BG analyzed the *SCL* genomic sequences and drafted parts of the manuscript. SB contributed ideas to CHAOS development and rewrote portions of the manuscript. BM implemented the new version of DIALIGN and drafted parts of the manuscript. All authors read and approved the final manuscript.

Acknowledgements

The authors would like to thank Chuong B. Do for help with CHAOS development, Nadine Werner for assistance with the manuscript, and Inna Dubchak for valuable conversations during this study. Rasmus Steinkamp developed the WWW interface for the CHAOS/DIALIGN software at GoBiCS. MB is supported by the NSF Graduate Research Fellowship. MC and BG are supported by the Wellcome Trust and the Leukaemia Research Fund. The work is partly supported by DFG grant MO 1048/1-1.

References

1. Miller W: **Comparison of genomic DNA sequences: solved and unsolved problems.** *Bioinformatics* 2001, **17**:391-397.
2. Frazer KA, Elnitski L, Church DM, Dubchak I, Hardison RC: **Cross-species sequence comparisons: A review of methods and available resources.** *Genome Research* 2003, **13**:1-12.
3. Chain P, Kurtz S, Ohlebusch E, Slezak T: **An applications-focused review of comparative genomics tools: capabilities, limitations, and future challenges.** *Briefings in Bioinformatics* 2003, **4**:105-123.
4. Gelfand MS, Mironov AA, Pevzner PA: **Gene recognition via spliced sequence alignment.** *Proc Natl Acad Sci USA* 1996, **93(17)**:9061-9066.
5. Bafna V, Huson DH: **The conserved exon method for gene finding.** *Bioinformatics* 2000, **16**:190-202.
6. Batzoglou S, Pachter L, Mesirov JP, Berger B, Lander ES: **Human and mouse gene structure: comparative analysis and application to exon prediction.** *Genome Research* 2000, **10(7)**:950-958.
7. Korf I, Flicek P, Duan D, Brent MR: **Integrating genomic homology into gene structure prediction.** *Bioinformatics* 2001:S140-S148.
8. Wiehe T, Gebauer-Jung S, Mitchell-Olds T, Guigó R: **SGP-1: Prediction and validation of homologous genes based on sequence alignments.** *Genome Research* 2001, **11**:1574-1583.
9. Rinner O, Morgenstern B: **AGenDA: Gene prediction by comparative sequence analysis.** In *Silico Biology* 2002, **2**:195-205.
10. Morgenstern B, Rinner O, Abdeddaïm S, Haase D, Mayer K, Dress A, Mewes H-W: **Exon discovery by genomic sequence alignment.** *Bioinformatics* 2002, **18**:777-787.
11. Hardison R, Slightom JL, Gumucio DL, Goodman M, Stojanovic N, Miller W: **Locus control regions of mammalian β-globin gene clusters: combining phylo-genetic analyses and experimental results to gain functional insights.** *Gene* 1998, **205**:73-94.
12. Jareborg N, Birney E, Durbin R: **Comparative analysis of noncoding regions of 77 orthologous mouse and human gene pairs.** *Genome Research* 1999, **9**:815-824.
13. Loots GG, Locksley RM, Blankespoor CM, Wang ZE, Miller W, Rubin EM, Frazer KA: **Identification of a coordinate regulator of interleukins 4, 13, and 5 by cross-species sequence comparisons.** *Science* 2000, **288(5463)**:136-140.
14. Göttgens B, Barton LM, Gilbert JGR, Bench AJ, Sanchez MJ, Bahn S, Mistry S, Grafham D, McMurray A, Vaudin M, Amaya E, Bentley DR, Green AR: **Analysis of vertebrate SCL loci identifies conserved enhancers.** *Nature Biotechnology* 2000, **18**:181-186.
15. Göttgens B, Barton L, Chapman M, Sinclair A, Knudsen B, Grafham D, Gilbert J, Rogers J, Bentley DR, Green AR: **Transcriptional regulation of the stem cell leukemia gene (SCL) comparative analysis of five vertebrate SCL loci.** *Genome Res* 2002, **12**:749-759.
16. Göttgens B, Gilbert JGR, Barton LM, Grafham D, Rogers J, Bentley DR, Green AR: **Long-range comparison of human and mouse SCL loci: localized regions of sensitivity to restriction endonucleases correspond precisely with peaks of conserved noncoding sequences.** *Genome Res* 2001, **11**:87-97.
17. Dieterich C, Wang H, Rateitschak K, Krause A, Vingron M: **Annotating regulatory DNA based on man-mouse genomic comparison.** *Bioinformatics* 2002, **18**:S84-S90.
18. Fitch JP, Gardner SN, Kuczmarski TA, Kurtz S, Myers R, Ott LL, Slezak TR, Vitalis EA, Zemla AT, McCready PM: **Rapid Development of Nucleic Acid Diagnostics.** *Proceedings of the IEEE* 2002, **90**:1708-1721.
19. Delcher LA, Kasif S, Fleischmann AD, Peterson J, White O, Salzberg SL: **Alignment of whole genomes.** *Nucleic Acids Res* 1999, **27(11)**:2369-2376.
20. Kurtz S, Schleiermacher C: ***REPuter*: Fast computation of maximal repeats in complete genomes.** *Bioinformatics* 1999, **15(5)**:426-427.
21. Kurtz S, Ohlebusch E, Schleiermacher C, Stoye J, Giegerich R: **Computation and visualization of degenerate repeats in complete genomes.** In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology Menlo Parc, CA, AAAI Press*; 2000:228-238.
22. Schwartz S, Zhang Z, Frazer KA, Smit A, Riemer C, Bouck J, Gibbs R, Hardison R, Miller W: **PipMaker–a web server for aligning two genomic DNA sequences.** *Genome Research* 2000, **10**:577-586.
23. Schwartz S, Kent WJ, Smit A, Zhang Z, Baertsch RHR, Haussler D, Miller W: **Human-mouse alignments with BLASTZ.** *Genome Research* 2003, **13**:103-107.

24. Morgenstern B: **DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment.** *Bioinformatics* 1999, 15:211-218.
25. Morgenstern B, Atchley WR: **Evolution of bhlh transcription factors: modular evolution by domain shuffling?** *Mol Biol Evol* 1999, 16:1654-1663.
26. Morgenstern B: **A simple and space-efficient fragment-chaining algorithm for alignment of DNA and protein sequences.** *Applied Mathematics Letters* 2002, 15:11-16.
27. Gusfield D: *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology Cambridge, UK: Cambridge University Press*; 1997.
28. Brudno M, Morgenstern B: **Fast and sensitive alignment of large genomic sequences.** In *Proceedings IEEE Computer Society Bioinformatics Conference: 14 – 16 August 2002; Paolo Alto* Edited by: *Vicky Markstein and Peter Markstein. IEEE Computer Society*; 2002:138-147.
29. Morgenstern B, Dress AWM, Werner T: **Multiple DNA and protein sequence alignment based on segment-to-segment comparison.** *Proc Natl Acad Sci USA* 1996, 93:12098-12103.
30. Abdeddaïm S, Morgenstern B: **Speeding up the DIALIGN multiple alignment program by using the 'greedy alignment of biological sequences library' (GABIOS-LIB).** *Lecture Notes in Computer Science* 2001, 2066:1-11.
31. McClure MA, Vasi TK, Fitch WM: **Comparative analysis of multiple protein-sequence alignment methods.** *Mol Biol Evol* 1994, 11:571-592.
32. Thompson JD, Plewniak F, Poch O: **BAliBASE: A benchmark alignment database for the evaluation of multiple sequence alignment programs.** *Bioinformatics* 1999, 15:87-88.
33. Lassmann T, Sonnhammer ELL: **Quality assessment of multiple alignment programs.** *FEBS Letters* 2002, 529:126-130.
34. Begley CG, Green AR: **The SCL gene: from case report to critical hematopoietic regulator.** *Blood* 1999, 93:2760-2770.
35. Barton LM, Göttgens B, Gering M, Gilbert JG, Grafham D, Rogers J, Bentley D, Patient R, Green AR: **Regulation of the stem cell leukemia (SCL) gene: a tale of two fishes.** *Proc Natl Acad Sci USA* 2001, 98:6747-6752.
36. Bockamp EO, McLaughlin F, Göttgens B, Murrell AM, Elefanty AG, Green AR: **Distinct mechanisms direct SCL/TAL-1 expression in erythroid cells and CD34 positive primitive myeloid cells.** *J Biol Chem* 1997, 272:8781-8790.
37. Bockamp EO, McLaughlin F, Murrell AM, Göttgens B, Robb L, Begley CG, Green AR: **Lineage-restricted regulation of the murine SCL/TAL-1 promoter.** *Blood* 1995, 86:1502-1514.
38. Sinclair AM, Göttgens B, Barton LM, Stanley ML, Pardanaud L, Klaine M, Bahn MGS, Sanchez M, Bench AJ: **Distinct 5' SCL enhancers direct transcription to developing brain, spinal cord, and endothelium: neural expression is mediated by GATA factor binding sites.** *Dev Biol* 1999, 209:128-142.
39. Lecointe N, Bernard O, Naert K, Joulin V, Larsen JC, Romeo PH, Mathieu-Mahul D: **GATA-and SP1-binding sites are required for the full activity of the tissue-specific promoter of the TAL-1 gene.** *Oncogene* 1994, 9:2623-2632.
40. Hyde-DeRuyscher RP, Jennings E, Shenk T: **DNA binding sites for the transcriptional activator/repressor YY1.** *Nuc Acids Res* 1995, 23:4457-4465.
41. Bray N, Pachter L: **MAVID multiple alignment server.** *Nucleic Acids Research* 2003, 31:3525-3526.
42. Brudno M, Do C, Cooper G, Kim MF, Davydov E, NISC Sequencing Consortium, Green ED, Sidow A, Batzoglou S: **LAGAN and multi-LAGAN: Efficient tools for large-scale multiple alignment of genomic DNA.** *Genome Research* 2003, 13:721-731.
43. Dubchak I, Brudno M, Loots GG, Pachter L, Mayor C, Rubin EM, Frazer KA: **Active conservation of noncoding sequences revealed by three-way species comparisons.** *Genome Research* 2000, 10:1304-1306.
44. Blanchette M, Schwikowski B, Tompa M: **Algorithms for phylogenetic footprinting.** *Journal of Computational Biology* 2002, 9:211-223.
45. Blanchette M, Tompa M: **Discovery of regulatory elements by a computational method for phylogenetic footprinting.** *Genome Research* 2002, 12:739-748.
46. Taher L, Rinner O, Gargh ASS, Brudno M, Batzoglou S, Morgenstern B: **AGenDA: Homology-based gene prediction.** *Bioinformatics* 2003, 19:1575-1577.
47. Altschul SF, Gish W, Miller W, Myers E-M, Lipman DJ: **Basic local alignment search tool.** *J Mol Biol* 1990, 215:403-410.
48. Pearson WR, Lipman DJ: **Improved tools for biological sequence comparison.** *Proc Natl Acad Sci USA* 1988, 85:2444-2448.
49. Bergman CM, Kreitman M: **Analysis of conserved noncoding dna in drosophila reveals similar constraints in intergenic and intronic sequences.** *Genome Research* 2001, 11:1335-1345.
50. Cioffi CC, Middleton DL, Wilson MR, Miller NW, Clem LW, Warr GW: **An IgH enhancer that drives transcription through basic helix-loop-helix and Oct transcription factor binding motifs. Functional analysis of the E(mu)3' enhancer of the catfish.** *J Biol Chem* 2001, 276:27825-27830.
51. Stoye J, Evers D, Meyer F: **Rose: Generating sequence families.** *Bioinformatics* 1998, 14:157-163.
52. Aho A, Corasick M: **Efficient string matching: an aid to bibliographic search.** *Comm ACM* 1975, 18:333-340.
53. Fredkin E: **Trie memory.** *Comm ACM* 1960, 3:490-500.
54. Pugh W: **Skip lists: A probabilistic alternative to balanced trees.** *Comm ACM* 1990, 33:668-676.